Saarland University

Simulated Penetration Testing and Mitigation Analysis

Dissertation
zur Erlangung des Grades
des Doktors der Ingenieurwissenschaften
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

von
Patrick Speicher

Saarbrücken, 2022

Tag des Kolloquiums:	18. Oktober 2022
Dekan:	Prof. Dr. Jürgen Steimle

Prüfungsausschuss:

Vorsitzender:	Prof. Bernd Finkbeiner, PhD
Berichterstattende:	Prof. Dr. Michael Backes
	Prof. Dr. Jörg Hoffmann
	Prof. Dr. Simon Hacks
Akademischer Mitarbeiter:	Dr. Robert Künnemann

Zusammenfassung

Da Unternehmensnetzwerke und Internetdienste stetig komplexer werden, wird es immer schwieriger, installierte Programme, Schwachstellen und Sicherheitsprotokolle zu überblicken. Die Idee hinter simuliertem Penetrationstesten ist es, Informationen über ein Netzwerk in ein formales Modell zu transferiern und darin einen Angreifer zu simulieren. Diesem Modell fügen wir einen Verteidiger hinzu, der mittels eigener Aktionen versucht, die Fähigkeiten des Angreifers zu minimieren. Dieses zwei-Spieler Handlungsplanungsproblem nennen wir *Stackelberg planning*. Ziel ist es, Administratoren, Penetrationstestern und der Führungsebene dabei zu helfen, die Schwachstellen großer Netzwerke zu identifizieren und kosteneffiziente Gegenmaßnahmen vorzuschlagen.

Wir schaffen in dieser Dissertation erstens die formalen und algorithmischen Grundlagen von Stackelberg planning. Indem wir dabei auf klassischen Planungsproblemen aufbauen, können wir von gut erforschten Heuristiken und anderen Techniken zur Analysebeschleunigung, z.B. symbolischer Suche, profitieren. Zweitens entwerfen wir einen Formalismus für Privilegien-Eskalation und demonstrieren die Anwendbarkeit unserer Simulation auf lokale Computernetzwerke. Drittens wenden wir unsere Simulation auf internetweite Szenarien an und untersuchen die Robustheit sowohl der E-Mail-Infrastruktur als auch von Webseiten. Viertens ermöglichen wir mittels webbasierter Benutzeroberflächen den leichten Zugang zu unseren Tools und Analyseergebnissen.

Abstract

As corporate networks and Internet services are becoming increasingly more complex, it is hard to keep an overview over all deployed software, their potential vulnerabilities, and all existing security protocols. Simulated penetration testing was proposed to extend regular penetration testing by transferring gathered information about a network into a formal model and simulate an attacker in this model. Having a formal model of a network enables us to add a defender trying to mitigate the capabilities of the attacker with their own actions. We name this two-player planning task *Stackelberg planning*. The goal behind this is to help administrators, penetration testing consultants, and the management level at finding weak spots of large computer infrastructure and suggesting cost-effective mitigations to lower the security risk.

In this thesis, we first lay the formal and algorithmic foundations for Stackelberg planning tasks. By building it in a classical planning framework, we can benefit from well-studied heuristics, pruning techniques, and other approaches to speed up the search, for example symbolic search. Second, we design a theory for privilege escalation and demonstrate the applicability of our framework to local computer networks. Third, we apply our framework to Internet-wide scenarios by investigating the robustness of both the email infrastructure and the web. Fourth, we make our findings and our toolchain easily accessible via web-based user interfaces.

Background of This Dissertation

This dissertation is based on the papers mentioned in the following. I contributed to all papers as one of the main authors.

Planning Algorithmic Background

[PSSB$^+$18] presented in Chapter 3 lays the algorithmic foundations for the rest of the thesis. It justifies our decision to focus solely on classical planning tasks s.t. probabilities are encoded as action costs instead of utilizing probabilistic formalism like MDP or POMDP. We formalize Stackelberg planning as a two-fold planning tasks s.t. the leader player maximizes the cost of the follower player while at the same time minimizing their own cost. Further, we introduce an algorithm for solving this kind of planning tasks in which we can utilize well-known techniques from the planning community and evaluate the performance on several well-known planning benchmark domains. Simulated penetration testing can be encoded as a two-player Stackelberg planning task where the defender is the leader player and the attacker is the follower player. The solution to such a task is the Pareto frontier of all cost-effective mitigation strategies. This paper was published at AAAI'18. I was mainly responsible for this work while Marcel Steinmetz gave crucial guidance for the algorithmic implementation and experiments. All authors contributed with their valuable ideas, general writing tasks, and reviews of the paper.

Additionally, in [PTSK$^+$21a] presented in Chapter 4, we improved the performance of solving Stackelberg planning tasks for some domains by utilizing symbolic search techniques. We introduce Symbolic Leader Search (SLS), a new algorithm for solving Stackelberg planning tasks. SLS aims to effectively reuse as much information as possible between the different subtasks. We achieve this by 1) recognizing that most follower subtasks can be solved with bounded cost suboptimal planning techniques, which are often more efficient than optimal planning algorithms, and by 2) using a symbolic representation to share information across searches. Our empirical evaluation shows that SLS outperforms our previous implementation for Stackelberg planning in many instances. The improvement is consistent, and is particularly pronounced in tasks with large leader action spaces. This paper was published at AAAI'21. Álvaro Torralba was mainly responsible for this work while I provided assistance for the implementation and evaluation. All authors contributed with their valuable ideas, general writing tasks, and reviews of the paper.

Stackelberg Planning Application: Local Computer Network

In [PSBK$^+$22] presented in Chapter 5, we developed a practical framework for applying Stackelberg planning to local computer networks. We first provide the theory for formally modeling privileges, credentials, and privilege escalations within (nested) access control mechanisms (ACM) in simulated penetration testing Stackelberg tasks. This theory allows us to classify known vulnerabilities in the National Vulnerability Database (NVD) w.r.t. their effect regarding the escalation of privileges for the attacker. A vulnerability can either provide 1) no escalation of privileges, 2) a local escalation in the same ACM, or 3) a hyper escalation gaining privileges in the ACM one step higher in the hierarchy. We further explain how one can classify large sets of vulnerabilities in a practical semi-automatic way, provide a toolchain for scanning large computer networks, solving the corresponding simulated penetration task and provide a proof-of-concept evaluation of the toolchain. This work is currently not yet published. I was mainly responsible for this paper, especially for the formalization of privilege escalation, while Guido Battiston implemented the feasibility study as well as the toolchain and executed the evaluation. All authors contributed with their valuable ideas, general writing tasks, and reviews of the paper.

Stackelberg Planning Application: Internet-Scale

We further investigated Internet wide potential areas of application for Stackelberg planning tasks. The first scenario from [PSSK$^+$18a] presented in Chapter 6 tackles the problem of intelligence agencies of specific countries or groups of countries spying on the email users of another country. This scenario can be modeled as a Stackelberg planning task where the attackers are the spying security agencies and the defenders are the email providers wanting to protect their users. We analyze the impact and cost efficiency of different known mitigation strategies and assess how protocols like IPsec, DNSSEC, DANE, SMTP STS, SMTP over TLS and other mitigation techniques like server relocation can be combined to improve the confidentiality of the email users. Finally, we provide an extensive case study for 45 combinations of attacker and defender countries. This work was published at EuroS&P'18. I was mainly responsible for this work while Marcel Steinmetz assisted in developing the algorithmic implementation and Milivoj Simeonovski conducted the data acquisition. All authors contributed with their valuable ideas, general writing tasks, and reviews of the paper.

While our first Internet wide scenario examined the security of emails, the second scenario from [PTSS$^+$22] presented in Chapter 7 studies the web infrastructure. It tackles nation-sponsored agencies, large infrastructure providers, as well as independent hacker groups trying to compromise web users via, e.g., DNS poisoning, certificate spoofing, and inclusion of malicious third-party JS code. We analyze the impact and cost efficiency of a multitude of proposals from IPsec over DNSSEC to TLS, SRI & co. Creating a suitable Stackelberg planning model and solving it with a newly developed graph-based algorithm allows us to analyze the infrastructure of the Alexa top 5000 domains. We discover that some infrastructure providers have a comparable threat potential to nations, but also find that a considerable increase of security is possible for relatively low cost due to the effectiveness of cheap mitigations at the application and transport layer. This work is currently under submission at S&P'22. Giorgio Di Tizio and I were jointly responsible for this paper. I focused on the graph-based implementation and evaluation while Giorgio Di Tizio and Milivoj Simeonovski were responsible for the data acquisition part. All authors contributed with their valuable ideas, general writing tasks, and reviews of the paper.

Prototypes

In [PTSK$^+$20] presented in Chapter 8, we paved way for users to play around with general planning and Stackelberg planning tasks in their browsers without installing the toolchain on their machines. We compiled the planner Fast Downward to WebASM and built a powerful GUI to 1) play around with the case study from [PSSK$^+$18a], and 2) solve arbitrary planning and Stackelberg planning tasks stored on the user's machine directly in the browser and displaying the results. Finally, we compared the performance of the native variant of Fast Downward to the WebASM variant on several benchmark domains and concluded that the performance impact is around 50% which is still acceptable and useful in practice. This work was published at ICAPS'20. I was mainly responsible for this work while Nicolas Tran implemented the compilation of Fast Downward to WebASM. All authors contributed with their valuable ideas, general writing tasks, and reviews of the paper.

The following Table 1 lists all scientific publications that I have co-authored, sorted by years.

Table 1: Co-authored publications, sorted by years. The highlighted ones are the underlying scientific publications of this thesis.

	Chapter	Publication
[PSBK$^+$22]	5	Speicher, P., Battiston, G., Künnemann, R., and Backes, M. Playing catss and maus: a theory of privilege escalations and applications to simulated pentesting. In *not published yet*, 2022. forthcoming
[PTSS$^+$22]	7	Tizio, G. D., Speicher, P., Simonovsky, M., Backes, M., Stock, B., and Künnemann, R. Pareto-optimal defenses for the web infrastructure: theory and practice. *ACM Transactions on Privacy and Security (TOPS)*, October 2022
[PTSK$^+$21a]	4	Torralba, Á., Speicher, P., Künnemann, R., Steinmetz, M., and Hoffmann, J. Faster stackelberg planning via symbolic search and information sharing. In Leyton-Brown, K. and Mausam, editors, *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI'21)*. AAAI Press, January 2021
[PTSK$^+$20]	8	Tran, N., Speicher, P., Künnemann, R., Backes, M., Torralba, A., and Hoffmann, J. Planning in the browser. In *System Demonstration at the 30th International Conference on Automated Planning and Scheduling (ICAPS'20)*, 2020
[SSSH$^+$19]		Speicher, P., Steinmetz, M., Hoffmann, J., Backes, M., and Künnemann, R. Towards automated network mitigation analysis. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 1971–1978, 2019
[SFGS$^+$18]		Fickert, M., Gnad, D., Speicher, P., and Hoffmann, J. Saarplan: combining saarland's greatest planning techniques. In *IPC 2018 planner abstracts*, 2018
[PSSK$^+$18a]	6	Speicher, P., Steinmetz, M., Künnemann, R., Simeonovski, M., Pellegrino, G., Hoffmann, J., and Backes, M. Formally reasoning about the cost and efficacy of securing the email infrastructure. In *Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P'18)*, 77–91, 2018
[PSSB$^+$18]	3	Speicher, P., Steinmetz, M., Backes, M., Hoffmann, J., and Künnemann, R. Stackelberg planning: towards effective leader-follower state space search. In McIlraith, S. and Weinberger, K., editors, *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)*, 6286–6293. AAAI Press, February 2018

Acknowledgments

I am deeply grateful to my advisor Prof. Michael Backes for giving me the amazing opportunity to pursue my Ph.D. in his group, his mentoring, guidance, and encouragements. He creates an aspiring workplace and always motivates to bring out the best of his students. I also like to thank Prof. Jörg Hoffmann for his help and mentoring since my bachelor's degree, invaluable advices, and for the inspiration to pursue a Ph.D. in the first place. Further, I wish to express my gratitude to Dr. Robert Künnemann whose door was always open for a chat, guided me from the start of my Ph.D. until its end, and who was happy to help every time I had questions, problems, or a need for a discussion. It was a pleasure working closely with you on all of our projects. Furthermore, I would like to give special credits to all my co-authors and colleagues, in particular, Guido Battiston, Milivoj Simeonovski, Marcel Steinmetz, Giorgio Di Tizio, Nicolas Tran, and Álvaro Torralba. You created a fun and fruitful working atmosphere and I thank you all for the interesting discussions about research and other questions of life. Most importantly, I want to thank all my family members for their unconditional support throughout all these years and always having my back. I would not stand where I stand today without them.

Contents

List of Figures

List of Tables

List of Algorithms

1
Introduction

1.1 State of the Art and Motivation

Network and web security, the defense against external attackers, is one of the central problems in the modern security landscape. Computer networks with Internet access are crucial for nearly every company offering some kind of Internet-related service or product, having multiple connected office locations, or with increasing importance, offering remote work capabilities for their employees. State-controlled critical infrastructure for, e.g., the power grid or the government are affected as well. As computer networks and web services are becoming increasingly more complex, it is hard to keep an overview over all deployed software, their potential vulnerabilities and all the possible security protocols and potential countermeasures. New vulnerabilities are found every single day and attacks on large companies or critical infrastructure are published on a weekly basis. It is hard to keep track with the dynamic security landscape and to guarantee the security of complex computer networks. Standard methods for reviewing the security, identifying vulnerabilities, and attack scenarios of networks are vulnerability scans and penetration testing. The former automatically identifies all known vulnerabilities on every host in the network and the latter describes the process of a human security expert simulating an attacker exploiting the found vulnerabilities. Scaling with increasingly large infrastructures is the core problem of this approach because it heavily relies on manual labor.

Simulated penetration testing was proposed to assist large-scale penetration testing by simulating an attacker based on a formal model of the network. The model can be semi-automatically generated from vulnerability scans, public vulnerability databases and manual inspection. The formal model is described in terms of hosts in the network and attacker actions which grant the attacker access over specific hosts if the respective preconditions are satisfied. Every instance has an initial state describing the status quo of the network and a goal condition describing the attacker's objective, e.g., the access to an important host.

AI planning methods have been proposed [BGH$^+$05; LSR10] for simulating the attacker in the model and computing the chain of actions, leading from the initial to a goal state. These approaches, which derive from earlier approaches based on attack graphs [PS98a; Sch99; SHJ$^+$02], assume complete knowledge over the network configuration, which is often unavailable to the modeler, as well as the attacker. There are also more recent approaches with Markov decisions processes (MDP) as the underlying state model to obtain a good middle ground between accuracy and practicality [DL14; Hof15] because it can represent the attacker's uncertainty of the action outcomes as a probability distribution. In this thesis, we tackle applications with and without uncertainty.

Having a formal model of a network makes it easy to add a second player to the simulation, i.e., a defender trying to mitigate the capabilities of the attacker with their own defender action. This allows for analyzing and comparing different mitigation strategies in terms of the (hypothetical) network resulting from their application. Algorithmically, the attacker-planning problem now becomes part of a larger two-player planning problem, in which the best mitigation strategies are computed. This min-max notion is similar to a Stackelberg game, which is frequently used in security games [KYK$^+$11]. That is why we named this kind of two-player planning task *Stackelberg planning*. The foundational assumption is that the leader/defender acts first, while the follower/attacker can choose its best response after observing this choice, similar to a market leader and their followers. The solution to a Stackelberg planning task is the set of all Pareto-optimal defender attacker strategies.

In this dissertation, we first lay the formal and algorithmic foundations for Stackelberg planning tasks in Part I. By building it in a classical planning framework, we can benefit from well-studied advanced heuristics, pruning techniques and other approaches to speed up the search. One of these approaches is *symbolic search* [McM93; EK09; TAK$^+$17]. Using this technique, we introduce Symbolic Leader Search (SLS), another algorithm for solving Stackelberg planning tasks. SLS aims to effectively reuse as much information as possible between the different follower subtasks and by thus outperforms our previous implementation in many instances. Then, we show two major areas for practical application of Stackelberg planning. The first area (see Part II) is large computer networks for which we designed a theory for privilege escalation and developed a holistic toolchain to assist the work of penetration testers. The second area of application (see Part III) is the robustness of large Internet infrastructures, namely the email infrastructure and the web infrastructure via, e.g., JavaScript inclusions, attacks via routing, compromise of name servers and certificate authorities. We assume that administrators, security engineers, and the management level of large email and web service companies want to protect the security and privacy of their users against hacker groups and intelligence agencies of foreign countries, on the one hand. On the other hand, these providers need to handle the trade-off between gained security and cost of deploying additional security protocols. There are not only costs for deploying the security protocols itself, but also for the potential loss of users because, e.g., their browsers are incompatible or the user experience takes a hit. This attack and defense trade-off scenarios are a perfect application for Stackelberg planning which allows us to suggest the best cost-effective countermeasures to lower the security risk for real world users. Lastly, we made our findings and our toolchain easily accessible for everyone via web-based user interfaces in Part IV. Our websites allow everyone to 1) play around with the case studies of Chapter 6 and Chapter 7 and 2) solve arbitrary planning and Stackelberg planning tasks stored on their machine directly in the browser and displaying the results.

1.2 Thesis Structure and Contributions

The contribution of this thesis consists of the following six chapters.

Algorithmic Background: Stackelberg Planning

Inspired by work on Stackelberg security games, in Chapter 3, we present Stackelberg planning [PSSB$^+$18], where a *leader* player in a classical planning task chooses a minimum-cost action sequence aimed at maximizing the plan cost of a *follower* player in the state resulting from applying the leader actions. As an example, the leader sets up network defenses and the follower's goal is to compromise the network including the defenses. Stackelberg planning can provide useful analyses not only in planning-based security applications like network penetration testing, but also to measure robustness against perturbances in more traditional planning applications (e.g. with a leader sabotaging road network connections in transportation-type domains). To identify all equilibria – exhibiting the leader's own-cost-vs.-follower-cost trade-off – we design *leader-follower search*, a state space search at the leader level which calls in each state an optimal planner at the follower level. We devise simple heuristic guidance, branch-and-bound style pruning, and partial-order reduction techniques for this setting. We run experiments on Stackelberg variants of IPC and pentesting benchmarks. In several domains, Stackelberg planning is quite feasible in practice.

Algorithmic Background: Symbolic Search and Information Sharing

Stackelberg planning naturally captures security-related (leader=defender, follower=attacker) as well as robustness-related (leader=adversarial event, follower=agent) scenarios. Solving such tasks requires solving many related planning tasks at the follower level (in the worst case, one for every possible leader plan). In [PTSK$^+$21a], which is presented in Chapter 4, we introduce new methods to tackle this source of complexity, through sharing information across follower tasks. Our evaluation shows that these methods can significantly reduce both the time needed to solve follower tasks and the number of follower tasks that need to be solved in the first place.

Local Computer Networks: Privilege Escalations and Applications to Pentesting

Companies use risk estimation techniques to analyze their networks, prioritize problems and compute mitigation strategies. These techniques operate on a model of the network, which is acquired by scanning for known vulnerabilities from public databases. Naturally, information about the effect of an exploit is scarce, so most models feature adversaries that advance hop by hop. To our knowledge, there is no model that represents host-level privileges systematically. In [PSBK$^+$22] presented in Chapter 5, we present the first theory for modeling privilege escalations within (nested) access control contexts and demonstrate its use by implementing a fully comprehensive system for network scanning and risk analysis, called CATSS/MAUS. By generating a Stackelberg planning task, CATSS/MAUS can not only determine the vulnerabilities of a network, it can also propose countermeasures, specifically those that 'break a chain' of privilege escalations. Further, we use our theory to justify a categorization of vulnerabilities. Only one bit of information more affords a vast improvement in semantic clarity and practicability. We use machine learning techniques to show that this categorization can be computed automatically and with high accuracy.

Internet-Scale: Email Infrastructure

Security in the Internet has historically been added post-hoc, leaving services like email, which, after all, is used by 3.7 billion users, vulnerable to large-scale surveillance. For email alone, there is a multitude of proposals to mitigate known vulnerabilities, ranging from the introduction of completely new protocols to modifications of the communication paths used by big providers [AAL+05a; R A05; AAL+05b; Hof02; DH15; MRL+16; Res00]. Deciding which measures to deploy requires a deep understanding of the induced benefits, the cost and the resulting effects. In [PSSK+18a] presented in Chapter 6, we propose the first automated methodology for making formal deployment assessments. Our Stackelberg planning algorithm analyzes the impact and cost efficiency of different known mitigation strategies against an attacker in a formal threat model. This novel formalization of an infrastructure attacker includes routing, name resolution and application-level weaknesses. We apply the methodology to a large-scale scan of the Internet, and assess how protocols like IPsec, DNSSEC, DANE, SMTP STS, SMTP over TLS and other mitigation techniques like server relocation can be combined to improve the confidentiality of email users. We analyzed 45 combinations of attacker and defender countries and nine cost scenarios. This is the first deployment analysis for mitigation techniques at this scale.

Internet-scale: Web Infrastructure

The integrity of the content a user is exposed to when browsing the web relies on a plethora of non-web technologies and an extensive infrastructure of interdependent hosts, communication technologies, and trust relations. Incidents like the Chinese Great Cannon attack or the MyEtherWallet attack make it painfully clear: the security of end users hinges on the security of the surrounding infrastructure, e.g., routing, DNS, content delivery, and the PKI. There are many competing, but isolated proposals to increase security, from the network up to the application layer. So far, researchers had a focus on analyzing attacks and defenses on specific layers. We still lack an evaluation of how, given the status quo of the web, these proposals can be combined, how effective they are, and at what cost the increase of security comes. In [PTSS+22] presented in Chapter 7, we propose an analysis based on Stackelberg planning that considers a rich attacker model and a multitude of proposals from IPsec to DNSSEC and SRI. Our threat model considers the security of billions of users against attackers ranging from small hacking groups to nation-state actors. We developed a novel graph-based algorithm to solve this kind of Stackelberg task which scales to analyzing the infrastructure of the Top 5k Alexa domains. We discover that the security mechanisms currently deployed are ineffective and that some infrastructure providers have a comparable threat potential to nations. Nevertheless, we find a considerable increase of security (up to 13% protected web visits) is possible at relatively modest cost, due to the effectiveness of cheap mitigations at the application and transport layer, which dominate expensive infrastructure enhancements such as DNSSEC and IPsec.

Prototypes: Planning in the Browser

Traditionally, planning tools are designed as applications that users need to install, taking care of the specificities of their operating system, packages, etc. This often places a significant burden on users, especially due to the academic nature of the software. To remedy this situation, we propose for planning to jump onto the growing trend of light-weight software use. In [PTSK+20] presented in Chapter 8, we demonstrate the possibility to run planning tools directly in the browser. We used the emscripten framework to port Fast Downward into WebAssembly (WASM) code, which allows for in-browser execution at near-native speeds. This allows for in-browser PDDL editing and planner execution without the need for server-side resource management. Moreover, it opens up a host of new possibilities, ranging from the interactive presentation of planning-based analysis to new applications exploiting the availability of planning in browsers to perform user-centric analyses. One exemplary use case is the case study from Chapter 6.

2
Related Work on Model-Based Security Analysis

We concentrate on different general methods of attack modeling and countermeasure selection here, postponing application- or planning-specific related work to the respective chapters. We can divide the related approaches into the following categories: 1) game-theoretic, 2) Integer programming (IP)-based, and 3) graph-based.

2.1 Game-Theoretic Approaches

Stackelberg planning models a leader as well as a follower which can also be seen as defender and attacker. At first sight, it relates to more general game-theoretic security models in which there are also attackers and defenders. The most prominent application of such models thus far concerns physical infrastructures and defenses (e.g. [Tam11; SJY$^+$14]) which are completely different from the network security setting and target, e.g., physical security like deploying air marshals in planes. Another important difference to most game-theoretic models is that Stackelberg planning does not consider arbitrarily long exchanges of action and counteraction, but only a single such exchange: leader applies mitigations to the network infrastructure, follower attacks the network including the mitigations.

The work more closely related to ours is that by Durkota et al. [DLK$^+$15b; DLK$^+$15a; DLB$^+$15; DLK$^+$16]. Like our mitigation analysis, Durkota et al.'s work line considers a Stackelberg formulation of security testing and is placed in the network security context. Major differences lie in the attacker and defender models considered as part of the game. On the one hand, on the defender's side, Durkota et al.'s model is limited to the placement of honeypots, modeled as additional fake machines added to the network within *pools* of equivalent machines indistinguishable to the attacker. This is in contrast to our framework which allows classical planning defender actions, and can thus model complex infrastructure modifications. On the other hand, on the attacker's side, Durkota et al.'s model is more general than ours. It considers probabilistic attacker actions, with an execution semantics where attacker actions refer to machine pools, and a concrete machine is chosen randomly. The latter is needed to give meaning to the honeypot placement (if a honeypot happens to be chosen, the attack is stopped immediately). Such complexity is not required, however, in our framework, where deterministic attacker actions suffice for modeling purposes. Algorithmically, the consequence is that Durkota et al.'s work focuses on tackling the complexity of attack planning, while our algorithms focus on tackling the complexity of search through the space of defender-action strategies.

2.2 IP-Based Approaches

Extensive research has been done in formulating physical attack and defense problems as integer or multilevel mixed-integer programming models. The possible applications are, for example, allocating air defense missiles to incoming air targets[Kar08], critical infrastructure protection planning[SC08], and optimal power grid protection[YZZ14]. These scenarios are again very different from the network security setting and another main difference lies in the computational complexity. Stackelberg planning is a two-fold planning task and a single classical planning task is already PSPACE-hard. However, IP is NP-complete and thus not all planning tasks can be translated to an IP problem and planning is generally more expressive. Strictly speaking, if we would try to model the leader player of a Stackelberg game as an IP task, we could neither capture leader plans with non-trivial length bounds nor dependencies between leader actions. As an example, in a simulated pentesting problem in accordance to the formalization presented in Chapter 5, we would like to model a defender action for installing a physical firewall and other actions for configuring firewall rules which are only applicable after having installed one. It is straightforward to model this dependency in a general planning task, but impossible in an IP problem.

2.3 Graph-Based Approaches

Simulated penetration testing is rooted in the consideration of attack trees and attack graphs, first introduced by Philipps and Swiler [PS98a]. Attack trees (e.g., [Sch99; MO05]), a form of 'Graphical Security Models' [KKM+13], gives the security administrator an overview of possible attack scenarios. These are directed acyclic AND/OR graphs organizing known possible attacks into a top-down refinement hierarchy. The human user writes that hierarchy, and the computer analyzes how attack costs and probabilities propagate through the hierarchy.

Because of the acyclicity, attack trees are not as expressive as attack graphs which are not required to be acyclic. Attack graphs (see [LI05] for an overview) give an attack scenario overview as well, but at the same time express attack actions and combinations thereof. An attack graph represents the space of possible attacks as atomic attack actions, s.t. each action is described by a conjunctive precondition and postcondition over relevant properties of the system under investigation. Different variants of attack graphs provide analyses at different levels of complexity, ranging from simple threat overviews (e.g., [Sch99; TL00]) to a full state-space verification (e.g., [RA00a; SHJ+02]). A prominent middle ground between the two is the *monotonic* formulation – positive preconditions and postconditions only – that we employ here as well, where attackers keep gaining new assets, but never lose any assets during the course of the attack [TL00; AWK02; JNO05; OBM06; NEJ+09; GG09; KOE+09].

Further, the attack graph is intended as an analysis of threats that arise through the possible combinations of the attack actions. The search for combinations of actions, reaching some goal, corresponds to the creation of an attack plan from the perspective of an attacker or penetration tester. While this can be translated to a classical planning problem in which the attacker selects the best actions to reach its goal, this relation to the AI community went thus far relatively unnoticed. Because of the monotonic formulation - positive postcondtions only - attack graphs are as expressive and complex as delete-relaxed classical planning problems [Hof15].

The classic attack trees and graphs don't consider mitigations nor a defender. A more recent line of research considers attack-defense trees (ADTree) (e.g. [KMR+10; KKM+13; KW17; KW18; FW20]), not based on standard sequential decision-making formalisms. In [FW20], they first extract all reasonable attacker and defender strategies from an ADTree. Second, they transform the so-called defense semantics to an IP optimization problem which can then be solved with a given IP solver to get the set of non-dominated defense strategies. Stan et al. [SBE+21] performed an extensive evaluation and comparison between their approach, other ADTree-based works, and Stackelberg planning. Their approach models the attacker with an attack graph and computes the optimal defender strategy using an A*-based algorithm. Both, [FW20] and [SBE$^+$21], are more expressive than Stackelberg planning in the sense that they support custom optimization goals (the attacker success) or optimize the overall risk in the system, i.e., considering all of the attack paths to all of the attacker goals. In Stackelberg planning, on the other hand, we only optimize for the cheapest or most likely attack path to the specified attacker goal. Then again, both works only compute one optimal defender response for a specific budget, while Stackelberg planning computes all Pareto-optimal solutions and thus all defender strategies for all relevant budgets. Regarding performance, Stackelberg Planning was the fastest method in the survey of [SBE$^+$21]. It was compared to the approaches of Stan et al. [SBE$^+$21] and Kordy and Widel [KW17; KW18; FW20]. But again, the compared models differ in expressiveness which inherently limits the meaningfulness of a runtime comparison.

Part I

Planning Algorithms for Simulated Penetration Testing and Mitigation Analysis

Advances in algorithms and heuristics for Stackelberg planning and simulated penetration testing and mitigation analysis tasks

3
Stackelberg Planning

3.1 Motivation and Contributions

Stackelberg security games have been extremely successful in the formalization and solution of defensive tasks regarding physical infrastructures, such as patrolling an airport [Tam11]. In such games, the *leader* (the defender) moves first, followed by the *follower* (the attacker). A solution (a Stackelberg equilibrium) is a leader/follower move pair that minimizes the defender's loss given optimal attacker play. The same principle has recently been applied to planning-based network security penetration testing, short *pentesting* (e.g. [BGH$^+$05; LSR10; DL14; Hof15]): the defender chooses the placement of honeypots (fake hosts), in a way minimizing the expected reward of an attacker whose attack is thwarted in case it runs into a honeypot [DLK$^+$15b; DLK$^+$15a; DLK$^+$16]. In scenarios like these, the leader may benefit from randomizing their strategy.

Here, we introduce *Stackelberg planning* games, where each of the two players is modeled as a full-scale classical planning agent. We conjecture that Stackelberg planning can be used in various applications beyond simulated pentesting as a way to measure robustness against worst-case perturbances. For example, in transportation, the "leader" may be a saboteur, or a malicious environment, planning to disrupt the road network, while the "follower" may be the agent that plans transportation in a (damaged) network. Optimal leader behavior then captures minimal damage to the road network causing maximum transportation efficacy loss, thus measuring the road network's robustness against damage given the desired transportation objectives. Similar analyses can make sense, e.g., for warehouse robotics, Mars rovers, airport ground traffic, power supply networks, pipeline networks, greenhouse conveyor belts, etc. Interest in domains like these is amply reflected in the International Planning Competition (IPC) benchmarks. We thus focus on the case where the leader's strategy is pure and can be observed by the follower.

In our Stackelberg planning formalization, the leader and follower control different actions in a classical planning task with non-negative action costs. The follower has a goal; the leader does not have a goal and instead pursues the objective to maximize the follower's plan cost. Hence an equilibrium is a pair of leader/defender plans where the leader cannot decrease own cost without also decreasing optimal follower-plan cost. We aim at producing *all* such equilibria (akin to a Pareto frontier). We believe this is useful – compared to aggregating leader and follower costs into a single utility – as it exhibits the actual trade-off (software updates vs. data loss; damaged road segments vs. transportation cost; etc.). Different variants of Stackelberg planning may, of course, be useful too and are left for future research.

Our framework per se is somewhat novel in that prior work on planning games considered general self-interested planning agents each pursuing an own goal [BJV03; LKM07; BDE$^+$09].[1] Our main contribution is on the algorithmic and experimental side, however, tackling leader-follower move combinations where each "move" is a planning problem in its own right. We design *leader-follower search*, interleaving two search levels where each state in the leader level of the search spawns a planning problem at the follower level. Towards efficiency at the leader level, we devise simple heuristic guidance, branch-and-bound style pruning against incumbent solutions, as well as partial-order reduction techniques. We run experiments on Stackelberg variants of IPC and simulated penetration testing benchmarks. Expectedly, a crucial performance factor in Stackelberg planning is the number of leader actions considered. In several domains, reasonably large numbers are feasible in practice.

3.2 Stackelberg Planning

We define *Stackelberg planning tasks* following the classical STRIPS framework. A Stackelberg planning task is a tuple $\Pi = (\mathcal{P}, \mathcal{A}^L, \mathcal{A}^F, \mathcal{I}, \mathcal{G}^F)$ of a set of *facts* $\mathcal{P}$, a set of *leader actions* $\mathcal{A}^L$, a set of *follower actions* $\mathcal{A}^F$, an *initial state* $\mathcal{I} \subseteq \mathcal{P}$, and the *follower goal* $\mathcal{G}^F \subseteq \mathcal{P}$. We require that $\mathcal{A}^L \cap \mathcal{A}^F = \emptyset$, and we denote by $\mathcal{A} = \mathcal{A}^L \cup \mathcal{A}^F$ the set of all actions. A *state* of Π is a subset of facts $s \subseteq \mathcal{P}$. S denotes the set of all states. Every action $a \in \mathcal{A}$ is associated with a *precondition* $pre_a \subseteq \mathcal{P}$, an *add list* $add_a \subseteq \mathcal{P}$, a *delete list* $del_a \subseteq \mathcal{P}$, and a non-negative cost $c_a \in \mathbb{R}_0^+$. An action a is *applicable* in a state s if $pre_a \subseteq s$. In that case, the state resulting from applying a to s is $s[[a]] := (s \setminus del_a) \cup add_a$. An action sequence $\langle a_1, \dots, a_n \rangle$ is applicable in a state s if a_1 is applicable in s, and $\langle a_2, \dots, a_n \rangle$ is applicable in $s[[a_1]]$. The resulting state is denoted $s[[\langle a_1, \dots, a_n \rangle]]$. The cost of an action sequence is $c_{\langle a_1, \dots, a_n \rangle} = \sum_{i=1}^{n} c_{a_i}$.

The syntax and state transition semantics just specified is standard classical planning except for the partitioning of actions across the leader L and follower F. The particularities of our Stackelberg planning framework pertain to the task's solutions, which we define as follows.

A *leader strategy* is a sequence of leader actions π^L applicable in $\mathcal{I}$. We denote by $S^L \subseteq S$ the set of all states reachable from $\mathcal{I}$ through a leader strategy. Let $s \in S^L$ be any such state. The leader's *best move* to s is a leader strategy π^{L*} to s whose cost

[1] General game playing (e.g. [GLP05; LHG08; Thi10; HST12]) is also related, but more remotely still due to the modeling differences between planning vs. the logic-programming based Game Description Language.

is minimal among all leader strategies ending in s; we denote that cost by $L^*(s)$. A *follower strategy* in s is a sequence of follower actions π^F that is applicable in s, and that achieves the follower goal, i.e., $\mathcal{G}^F \subseteq s[[\pi^F]]$. The follower's *best response* in s is a follower strategy π^{F*} in s with minimal cost; we denote that cost by $F^*(s)$, or $F^*(s) = \infty$ if no follower strategy exists.

The leader's objective is to minimize their own cost L^*, while maximizing the cost F^* of the follower's best response. We capture the trade-off between these two objectives through the set of *equilibria* where L^* cannot be reduced without also reducing F^*. Precisely, we say that a state $s \in S^L$ is an equilibrium if it is not *dominated* by any other state $t \in S^L$, where t dominates s if $(L^*(t), F^*(t))$ dominates $(L^*(s), F^*(s))$, and a pair (L, F) dominates a pair (L', F') if $L \leq L'$ and $F \geq F'$ and at least one of these two inequalities is strict. The *solution* to a Stackelberg planning task is the set $S^* \subseteq S^L$ of all equilibria.

Observe that each equilibrium state s characterizes a subset of equilibrium strategy pairs, namely the pairs (π^{L*}, π^{F*}) of best move/response pertaining to s. In this sense, our definition of equilibria in terms of states is equivalent to one in terms of strategy pairs, but is more compact.

We remark that previous work on Stackelberg security games [Tam11; DLK$^+$15b], like the original definition of Stackelberg games [vSta34], defines overall utility as a function of both, the leader and follower strategies. An equilibrium is then a strategy pair where the follower strategy is a best response, and the overall leader utility is optimal among such strategy pairs. As discussed in the introduction, our definition aims instead at avoiding the aggregation of leader costs with follower costs, to exhibit the full trade-off S^* between these two functions.

3.3 Leader-Follower Search

As previously indicated, our base search algorithm, *leader-follower search*, is a two-layer search solving a classical planning task optimally in every node of a state space search at the leader level. We will devise pruning and partial-order reduction techniques in the following sections. The basic algorithm is depicted in Algorithm 3.1.

The algorithm is straightforward. It explores the entire leader state space, calling an optimal planner at the follower level for each candidate state. The only major addition is the maintenance of the incumbent solution $\hat{S}$: the subset of non-dominated search nodes found thus far. Dominance over search nodes here is defined in terms of $N.L$ and $F^*(N.s)$: the former replaces $L^*(N.s)$ which is (in general) not known during the search, while the latter is already cached in the respective search nodes at the time required.

The algorithm guarantees, for every $s \in S^L$, that eventually a search node N is expanded where $N.s = s$ and $N.L = L^*(s)$. Correctness follows immediately from that:

Proposition 1. *Upon termination of leader-follower search,* $\{\hat{N}.s \mid \hat{N} \in \hat{S}\} = S^*$.

Observe that, to attain this guarantee, the search is exhaustive: prior to termination, *every* state $s \in S^L$ is expanded at least once, regardless of the expansion order. However, a good expansion order is crucial for anytime behavior and for the efficacy of the pruning techniques we introduce below. (Near-)equilibria states should be found early on.

Algorithm 3.1: Leader-follower search. Search nodes N contain the state $N.s$ and the leader path cost $N.L$ to s. They also cache the follower's best-response cost $F^*(N.s)$.

```
procedure leader-follower-search($\Pi$)
Input: Stackelberg planning task $\Pi$
Output: set of equilibrium states $\{\hat{N}.s \mid \hat{N} \in \hat{S}\}$
$\hat{S} := \emptyset$;
let Open be an empty queue;
push node $N$ with $N.s := \mathcal{I}$ and $N.L := 0$ onto Open;
while Open $\neq \emptyset$ do
    pop $N$ from Open;
    // Follower-Search Pruning (Section 3.4)
    run an optimal planner to compute $F^*(N.s)$;
    if $N$ is not dominated by any $\hat{N} \in \hat{S}$ then
        remove from $\hat{S}$ every $\hat{N} \in \hat{S}$ dominated by $N$;
        $\hat{S} := \hat{S} \cup \{N\}$;
    for every $a \in \mathcal{A}^L$, $pre_a \subseteq N.s$ do
        // Partial-Order Reduction (Section 3.5)
        let $N'$ be a new search node;
        $N'.s := N.s[[a]]$; $N'.L := N.L + c_a$;
        if already generated $N'.s$ with path cost $\leq N'.L$ then
            continue
        // Leader-Search Pruning (next Section)
        push $N'$ onto Open;
return $\{\hat{N}.s \mid \hat{N} \in \hat{S}\}$;
```

The latter poses an interesting challenge to the generation of heuristic functions, namely a new type of estimation that one may baptize *Stackelberg heuristic functions*, based on relaxed Stackelberg planning problems that combine an optimistic estimation of leader costs with a pessimistic estimation of associated follower costs. There are at least two desirable outcomes of such estimation processes: a) for guiding leader-follower search, an estimate of the distance to the nearest equilibrium state; b) for admissible pruning, bounds on the leader/follower cost pairs still achievable below a given state. The comprehensive investigation of these questions has, in our view, the potential for an entire sub-area of AI Planning. We discuss b) in the next section, devising first pruning techniques; towards a) we implemented simple heuristics without any lookahead on the leader side, preferring leader states with higher follower plan cost estimates.

3.4 Branch-and-Bound Style Pruning

We define two pruning criteria for leader-follower search:

1. Follower-search pruning prunes unnecessary calls to the follower level, and can be done given the availability of an upper bound on follower cost.

2. Leader-search pruning prunes entire unnecessary search branches at the leader level. In order to identify such branches, bounds on the trade-off between leader and follower cost are required.

3.4.1 Follower-Search Pruning

Follower-search pruning in Algorithm 3.1 identifies N that are necessarily dominated by some $\hat{N} \in \hat{S}$. For such N, we do not need to compute $F^*(N.s)$. This is beneficial given the complexity of such a computation (optimal planning), and the number of times the computation will need to be performed without pruning (for every search node N).

To permit the desired pruning, it is enough to have an upper bound $\mathsf{up}^F(s)$ on the follower's best-response cost in a state s, i.e., $\mathsf{up}^F(s) \geq F^*(s)$. Namely, a given search node N is necessarily dominated by some $\hat{N} \in \hat{S}$ if N is dominated by $\hat{N}$ even under the optimistic assumption (from the leader's point of view) that $F^*(N.s) = \mathsf{up}^F(N.s)$.

Upper bounds $\mathsf{up}^F(s)$ can be computed, for example, by running a satisficing planner to compute any (not necessarily optimal) follower strategy. A much cheaper method, that we use in our experiments, is to derive $\mathsf{up}^F(s)$ from $F^*(t)$ for a parent state t of s, $s = t[[a]]$, using the following observation: If the follower's best-response π^{F*} in state t is not affected by a, i.e., π^{F*} still constitutes a follower strategy in s, then $F^*(s) \leq F^*(t)$, and hence we can use $\mathsf{up}^F(s) := F^*(t)$ as the upper-bound. If a does invalidate π^{F*}, we use the trivial upper bound $\mathsf{up}^F(s) = \infty$.

3.4.2 Leader-Search Pruning

Leader-search pruning is more ambitious than follower-search pruning. Rather than identifying nodes necessarily dominated by $\hat{S}$, it identifies nodes whose *descendant nodes* are necessarily dominated by $\hat{S}$. Given some node N' (the new search node in

Algorithm 3.1), we need to show that *for every descendant node N_0 of N', there exists a node $\hat{N} \in \hat{S}$ that dominates N_0*. If we succeed in showing this, then N' can be pruned.

Observe that assessing this property necessarily requires information about the trade-off between leader-cost vs. follower-cost, below N': How much does the leader still need to spend in order to increase follower cost, and by how much? Is it possible to obtain a new trade-off not dominated by any of the trade-offs currently contained in $\hat{S}$?

A conceptually simple approach could be derived through Stackelberg heuristic functions of type b) discussed in the previous section. Precisely, one could strive to design a function h^{Stackel} mapping N' to a set H of (L^+, F^+) pairs optimistically approximating the available trade-offs below N', in that, for every descendant node N_0 of N', there exists $(L^+, F^+) \in H$ that dominates $(N_0.L, F^*(N_0.s))$. Given such an estimation $h^{\text{Stackel}}(N') = H$, one can prune N' if, for every pair $(L^+, F^+) \in H$, there exists $\hat{N} \in \hat{S}$ such that $(\hat{N}.L, F^*(\hat{N}.s))$ dominates (L^+, F^+).

While natural, the computational feasibility of this approach appears questionable, as an entire Pareto frontier needs to be estimated. To derive a more feasible approach, we instead perform the assessment for the concrete incumbent solution $\hat{S}$, estimating whether or not it will be possible to beat these trade-offs. We formulate such estimation through an upper bound $\mathsf{up}^L(N', B)$ on the follower's best-response in any node reachable, for the leader, from N' within cost B. The advantage of this formulation is that suitable values for B can be gleaned directly from $\hat{S}$:

Theorem 1. *Every descendant of N' is dominated by some $\hat{N} \in \hat{S}$ if the following conditions are satisfied:*

(i) *$\hat{S}$ is not empty,*

(ii) $\mathsf{up}^L(N', \infty) \leq \max_{\hat{N} \in \hat{S}} F^*(\hat{N}.s)$,

(iii) $N'.L > 0$ *or* $\mathsf{up}^L(N', 0) < F^*(\hat{N}.s)$ *where* $\hat{N} \in \hat{S}$ *is such that* $\hat{N}.L = 0$, *and*

(iv) *for every $\hat{N}_1 \in \hat{S}$ with $\hat{N}_1.L > 0$ and $\hat{N}_1.L \geq N'.L$, there exists $\hat{N}_2 \in \hat{S}$ such that $\hat{N}_2.L < \hat{N}_1.L$ and $F^*(\hat{N}_2.s) \geq \mathsf{up}^L(N', \hat{N}_1.L - N'.L)$.*

Proof. Assume for contradiction that N' can reach a node N_0 that is not dominated by any $\hat{N} \in \hat{S}$, even though the conditions (i) – (iv) are all satisfied. If $\hat{S}$ is not empty, then observe that $\hat{S}$ is guaranteed to contain at least one node $\hat{N} \in \hat{S}$ with $\hat{N}.L = 0$. This holds because the algorithm in Algorithm 3.1 always adds the node with $N.s = \mathcal{I}$ and $N.L = 0$ to $\hat{S}$, and this node may only be replaced by some other node reached via a path with cost 0. If $N_0.L = 0$, then in particular $N'.L = 0$, and since N_0 is not dominated by $\hat{N}$, it holds that $F^*(N_0.s) \geq F^*(\hat{N}.s)$. However as (iii) is satisfied, either $N_0'.L > 0$ for all nodes N_0' reachable from N', or $F^*(N_0'.s) \leq \mathsf{up}^L(N', 0) < F^*(\hat{N}.s)$ for all nodes N_0' reachable from N' with $N_0'.L = 0$, i.e., N_0' is dominated by $\hat{N}$. Therefore, $N_0.L$ must be larger than 0. On the other hand, since (ii) is satisfied, $\hat{S}$ must contain a node $\hat{N}$ such that $F^*(N_0.s) \leq \mathsf{up}^L(N', \infty) \leq F^*(\hat{N}.s)$. Since N_0 is not dominated by any such $\hat{N}$, it immediately follows that $N_0.L \leq \hat{N}.L$. Let $\hat{N}_1 \in \hat{S}$ be the node with minimal $\hat{N}_1.L$ value among the nodes with $\hat{N}_1.L \geq N_0.L$. As we have shown before, $N_0.L > 0$, and therefore $\hat{N}_1.L > 0$. Because of (iv), there must hence exist $\hat{N}_2 \in \hat{S}$ such that $\hat{N}_2.L < \hat{N}_1.L$ and $F^*(\hat{N}_2.s) \geq \mathsf{up}^L(N', \hat{N}_1.L - N'.L) \geq F^*(N_0.s)$. However, the selection of $\hat{N}_1$ implies that $\hat{N}_2.L < N_0.L$. Thus, $\hat{N}_2$ actually dominates N_0, what is a contradiction to the assumption. We conclude that if N' can reach a node N_0 that is not dominated by any $\hat{N} \in \hat{S}$, then one of the conditions (i) – (iv) must be violated. □

At first glance, one might wonder why condition (iv) requires the consideration of two nodes $\hat{N}_1, \hat{N}_2 \in \hat{S}$. To see why, assume a node N' that can only reach a single node N_0 whose L and F^* values are both either strictly smaller or strictly larger than those of $\hat{N}$, for every $\hat{N} \in \hat{S}$. When considering only a single node $\hat{N} \in \hat{S}$ in the reachability approximation, all we will find out is that within a cost budget of $\hat{N}.L$, N' cannot reach any node where the follower's best-response cost is at least as large as $F^*(\hat{N}.s)$. Nevertheless, N_0 is not dominated by any $\hat{N} \in \hat{S}$, and thus N' should be considered for expansion.

As of now, we use a very simple upper bound $\mathsf{up}^L(N', B)$: let $\mathsf{up}^L(N', B) := \mathsf{up}^L_\forall$ where $\mathsf{up}^L_\forall$ is an upper bound on the follower's best-response in *every* state. For example, using the trivial such upper bound $\mathsf{up}^L_\forall := \infty$, as soon as $\hat{S}$ contains a node $\hat{N}$ with $F^*(\hat{N}.s) = \infty$, i.e., a state in which the follower can no longer achieve their goal, all nodes N' with $N'.L > \hat{N}.L$ will be pruned. To derive tighter upper bounds $\mathsf{up}^L_\forall$, in Stackelberg planning tasks where the leader can never entirely preclude the follower from reaching the goal, one can over-approximate the overall harm the leader can do to the follower. In our experiments, we use a simple technique based on the delete relaxation. We first perform a delete-relaxed reachability fixed point for the initial state, considering only leader actions. We collect the facts p deleted by at least one leader action applicable in the fixed point. We remove all these facts p from the initial state, and we compute a follower best-response (an optimal follower strategy) given this reduced initial state.

3.5 Partial-Order Reduction

As an additional means to reduce the leader search space, we adapt a well-known partial-order reduction technique, *strong stubborn sets (SSS)* [Val89; WH12; WHA+13; WH14]. The basic idea behind SSS is to exploit action independency to identify, for a given search state s, a subset A of applicable actions so that expanding s using only A suffices to retain the standard notions of completeness and optimality. Here, we adopt this idea to preserve the guarantee given by Proposition 1, i.e., that $\{N.s \mid N \in \hat{S}\} = S^*$ upon termination of leader-follower search.

The construction of an SSS follows three steps: (i) one starts with a *disjunctive action landmark*, i.e., a set of actions one of which is needed to achieve the goal; (ii) one backchains over open action preconditions to actions applicable in s; and (iii) for each applicable action one includes all interfering actions. (ii) and (iii) remain the same in our context, and we specify them formally below. The main issue here is (i), because in Stackelberg planning the goal of the leader is not given explicitly. So we need to find a different way to seed the SSS with "something the leader will definitely have to do in order to make progress".

To that end, say that $s = N.s$ is the leader search state about to be expanded in leader-follower search as per Algorithm 3.1. Say that π^{F*} is the follower's best-response in s. Consider any minimal-cost path $\mathcal{I} = s_0, a_1, s_1, \dots, a_n, s_n$ through $s = s_i$ to an equilibrium $s_n \neq s$. Observe that *either the path behind s_i must contain a leader action that renders π^{F*} inapplicable, or all actions on that path must have cost* 0*:* clearly, if both these conditions are false, then s dominates s_n in contradiction. We thus seed our SSS with the set of all leader actions that may disable π^{F*} and handle 0-cost actions separately.

In detail, we define SSS for leader-follower search as follows. Let $a_1, a_2 \in \mathcal{A}^L$, $a_1 \neq a_2$, be two leader actions. Following classical planning definitions of SSS, we say that a_1 *disables* a_2 if $pre_{a_2} \cap del_{a_1} \neq \emptyset$; a_1 *enables* a_2 if $add_{a_1} \cap pre_{a_2} \neq \emptyset$ and a_1 does not disable a_2; a_1 and a_2 *conflict* with each other if $add_{a_1} \cap del_{a_2} \neq \emptyset$ or vice versa $add_{a_2} \cap del_{a_1} \neq \emptyset$; a_1 and a_2 *interfere* if they conflict with each other, or either disables the other. Furthermore, we define regression in the usual manner, namely for a set of facts $G \subseteq \mathcal{P}$ and an action $a \in \mathcal{A}$, the regression of G through a is defined if $add_a \cap G \neq \emptyset$ and $del_a \cap G = \emptyset$. If defined, the regression is given by the set $\text{Regress}(G, a) = (G \setminus add_a) \cup pre_a$.

Now, $A \subseteq \mathcal{A}^L$ is an *SSS in s for* π^{F*} if the following conditions are satisfied:

(i) A contains every leader action $a \in \mathcal{A}^L$ where $del_a \cap \text{Regress}(\mathcal{G}^F, \pi^{F*}) \neq \emptyset$;

(ii) for every $a \in A$ not applicable in s, A contains all actions that enable a; and

(iii) for every $a \in A$ applicable in s, A contains all actions that interfere with a.

Note that $\text{Regress}(\mathcal{G}^F, \pi^{F*})$ gives the minimal set of facts that is required for the follower to achieve their goal through π^{F*}. Facts appearing e.g. in the precondition of some action a in π^{F*}, but not in $\text{Regress}(\mathcal{G}^F, \pi^{F*})$, are not relevant for the applicability of π^{F*} because, by definition of regression, these are added by some other action appearing before a anyway. Considering also such facts would not harm the correctness of the SSS construction, but may lead to larger sets A, and hence less pruning. With the same arguments as in classical planning [WH12], we obtain:

Theorem 2. *Let A be an SSS in s for π^{F*}. Then, for every leader strategy $a_1, \dots, a_n$ in s where π^{F*} is no longer applicable in $s[[a_1, \dots, a_n]]$, there exists $1 \leq i \leq n$ such that $a_i \in A$, and a_i, a_1, $\dots$, a_{i-1}, a_{i+1}, $\dots$, a_n is applicable in s, and $s[[a_1, \dots, a_n]] = s[[a_i,$ $a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n]]$.*

As an immediate consequence, we can ignore in the expansion of s all actions $a \in \mathcal{A}^L$ whose cost is larger than 0 and which are not contained in A.

3.6 Experiments

To evaluate the scalability of our Stackelberg planning algorithms, we extended several benchmark domains from the International Planning Competition (IPC). Due to the intrinsic difficulty of solving classical planning tasks and hence Stackelberg planning tasks optimally, we also report results for a *satisficing* variant of the leader-follower search algorithm, where we used an inadmissible search configuration to find follower strategies. Our implementation is based on Fast Downward [Hel06]. We ran all experiments on 2.20 GHz Intel Xeon machines, with runtime/memory limits of 30 min/4 GB (as typically used in IPC setups).

3.6.1 Benchmarks

To design our benchmarks, we adapted three domains relating to transportation, namely *Logistics* (IPC'98), *Nomystery* (IPC'11), and *Visitall* (IPC'14); the puzzle game *Sokoban* (IPC'14); and a simulated pentesting domain *Pentest* adapted from prior work on this application.

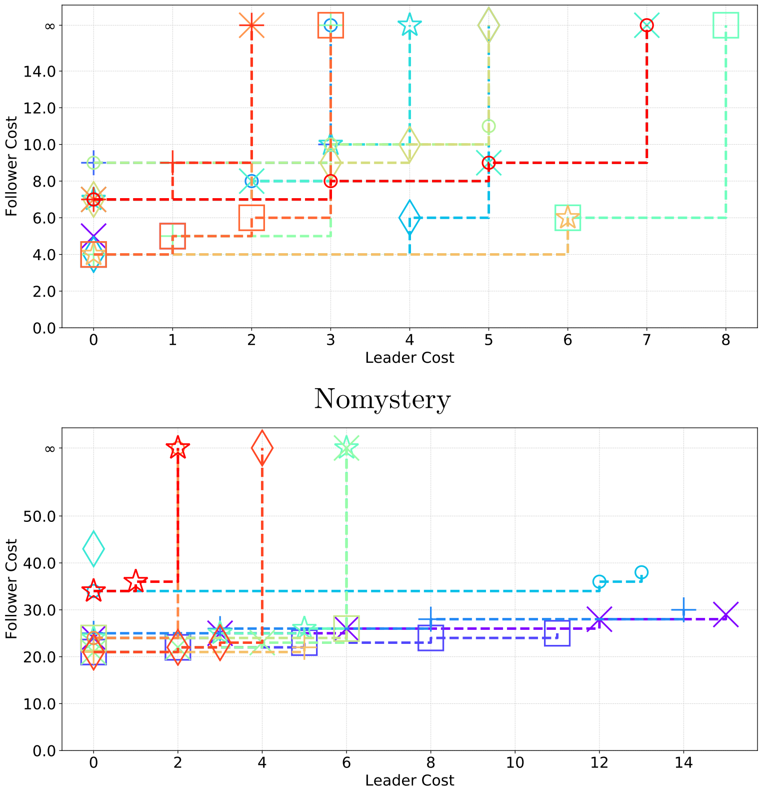
Given an individual classical planning base task (a benchmark domain instance), we obtain Stackelberg planning tasks as follows. We set the follower's actions and goal to be that of the base task. We design a set of leader actions suitable for the domain and task (see details below). To control the complexity stemming from the number of leader actions, we generate multiple Stackelberg planning instances, setting that number within $\{1, 2, 3, 4, 5, 6, 8, 10, 12, 16, 25, 32, 50, 64, 128, 256, 512, 1024, 2048, 4096\}$ up to the number of leader actions present in the task. We hence get up to 20 different Stackelberg planning instances per base task. In total, our benchmark suite contains 3263 instances.

In the three transportation-like domains, the leader actions are designed based on the idea of evaluating road-network robustness.[2] Any one leader action removes one road connection (an edge in the network graph) from the network. We design two different variants (that we will refer to by their number throughout this section): (i) the leader can execute any such action at any time; (ii) in order to remove the connection between X and Y, the leader must be located at either X or Y. In (ii), the leader can change their location via additional leader actions moving on the same road network; once a connection is removed, the leader cannot use it anymore, rendering even the leader planning problem on its own non-trivial, requiring to reason about interferences.

In Sokoban, we test for robustness against modifications to the board. Here, each leader action allows to block one cell of the grid by placing a wall. We again consider two variants: (i) there is no precondition to placing a wall; and (ii) walls may only be placed in a cell adjacent to other walls.

For generating the Pentest base instances, we adapted the MDP generator by Steinmetz et al. [SHB16] to output classical planning tasks. We applied two changes: (i) each exploit action is now guaranteed to succeed, and (ii) the cost of an exploit action is set to the negative logarithm of the exploit success probability. This changes the overall objective from the MDP problem of finding a *policy* maximizing the probability of achieving the goal, to the classical planning problem of finding an *action sequence* maximizing that probability. Steinmetz et al.'s generator allows to scale both the network size `N` and the number of exploits `E` in the network. We used `E` := `N` for simplicity, and scaled `N` from 50 to 600 in steps of 50. For each `N` we generated three instances using a different random seed, resulting in a total number of 36 base instances. We designed the leader actions to correspond to the activities of an administrator trying to secure the network. Namely, we consider two possible *fixes*: closing a specific port on a specific host as a firewall rule; and patching a software on a specific host, effectively removing the associated exploits.

[2] We remark on the side that this application was inspired by a road-network breakdown in our own city, blocking reasonable access to the university campus for 3 weeks.



Nomystery

Visitall

Figure 3.1: Visualization of Pareto frontiers in Nomystery and Visitall. Each line shows the Pareto frontier of an instance, each data point an entry in the frontier.

3.6.2 Pareto Frontier Examples

The solution for a Stackelberg planning instance is a Pareto frontier S^*, which visualizes the leader/follower-cost trade-off. To exemplarily illustrate that this S^* can truly be of interest, we show frontiers of several Nomystery and Visitall instances in Figure 3.1. We used variant (ii) for both domains, so the leader must be located at an adjacent location to remove a connection and there are leader move actions. Each individual line represents a complete frontier and every marker represents an entry in the frontier, i.e., a leader/follower-cost pair (L, F). One can observe many instances where it is possible to increase F to ∞ with very little L, but also instances where one cannot increase F at all, only by a small margin, or only with high L.

3.6.3 Optimal Stackelberg Planning

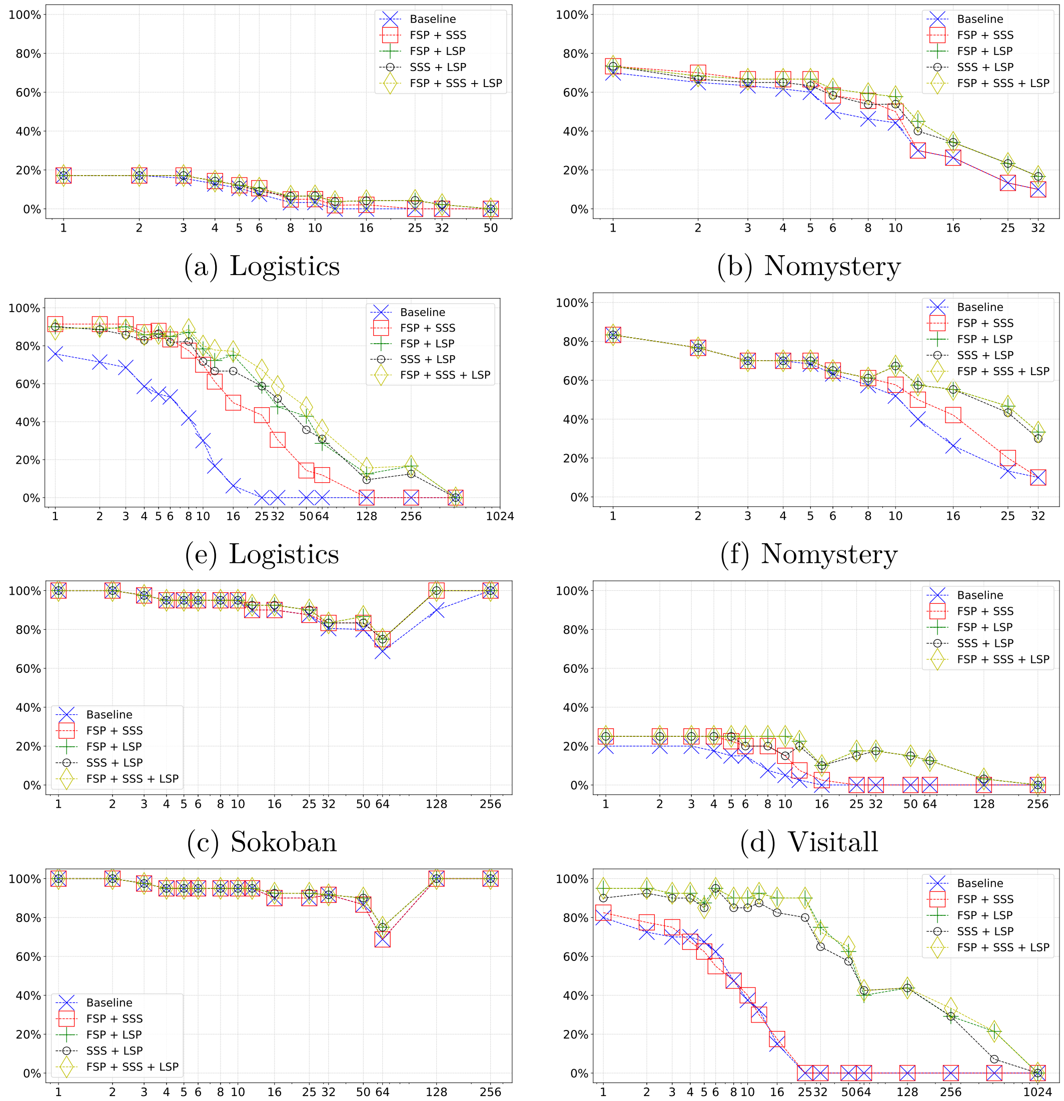
We next shed light on the feasibility of optimal Stackelberg planning, and on the impact of our pruning methods. To find optimal follower responses, we run A* with the state-of-the-art admissible heuristic LM-cut [HD09]. Due to the nature of Stackelberg planning, and our benchmarks in particular, there is a high chance to exclude all follower strategies. To be able to perform well in such cases – i.e., to effectively prove the follower problem unsolvable – we use, in addition to LM-cut, the PDB heuristic that was part of Aidos, the winning portfolio in the 2016 Unsolvability IPC [SPS$^+$16]. We noticed early on in our experiments that the PDB heuristic helps a lot in a few instances, although the overall impact is rather small.

At the leader level, we run iterative deepening depth-first search. We order open states by decreasing follower optimal response cost (preferring states with higher follower cost). If there is no follower strategy for a state and no 0-cost leader actions, then we can trivially discard this state. We refer to the branch-and-bound follower-search pruning, as discussed before, by *FSP*; to leader-search pruning based on the upper-bound $\mathsf{up}^L_\forall$ by *LSP*; and to our strong stubborn sets technique by *SSS*. We report results for different combinations of these techniques (including none, as a baseline) to examine their impact.

The coverage results are shown in Figure 3.2 and Figure 3.3. Note that we aggregated both types of leader actions in Figure 3.2, because the results were quite similar. In almost all Sokoban instances (Figure 3.2 (c)), few changes to the board suffice to prevent the follower from reaching their goal, yielding a shallow leader-level search tree. All our configurations tackle such cases effectively, and relative coverage is between 80% and 100% throughout. (The increase from 64 to 128 leader actions is an artifact of a decreasing number of instances for an increasing number of leader actions.)

In contrast, in Nomystery (Figure 3.2 (b)) and Pentest (Figure 3.3 (a)), the impact of the number of leader actions is large, as one would expect. In Logistics and Visitall (Figure 3.2 (a) and (d)), coverage is extremely low even with few leader actions. This is due to the computational cost of computing optimal follower responses, which is prohibitive in these two domains.

Comparing performance across different configurations, the coverage differences are small. The largest difference occurs in Pentest, where the baseline configuration without any pruning falls substantially short starting from 4 leader actions. The overall best-performing configuration is the one where all pruning techniques are enabled. The



(a) Logistics

(b) Nomystery

(e) Logistics

(f) Nomystery

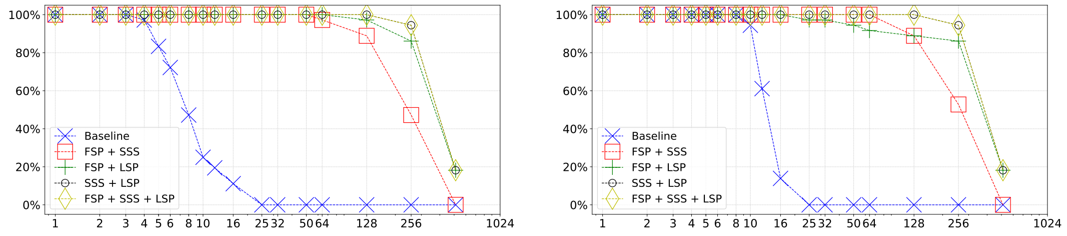
(c) Sokoban

(d) Visitall

(g) Sokoban

(h) Visitall

Figure 3.2: Percentage of solved instances, as a function of the number of blockable cells (Sokoban) respectively removable road-map connections (other domains). Plots (a) – (d) report the results for optimal Stackelberg planning, plots (e) – (h) for greedy Stackelberg planning.



(a)

(b)

Figure 3.3: Percentage of solved Pentest instances, as a function of the number of fixes. Optimal (a), greedy (b).

configuration without SSS performs almost equally well except in Pentest.

A closer look at the performance of the different configurations, beyond the coarse measurement afforded by coverage, shows that actually all our pruning methods yield vast runtime advantages over the baseline. Figure 3.4 shows per-instance runtime scatter plots. Clearly, runtime reductions by orders of magnitude are common for all three methods.

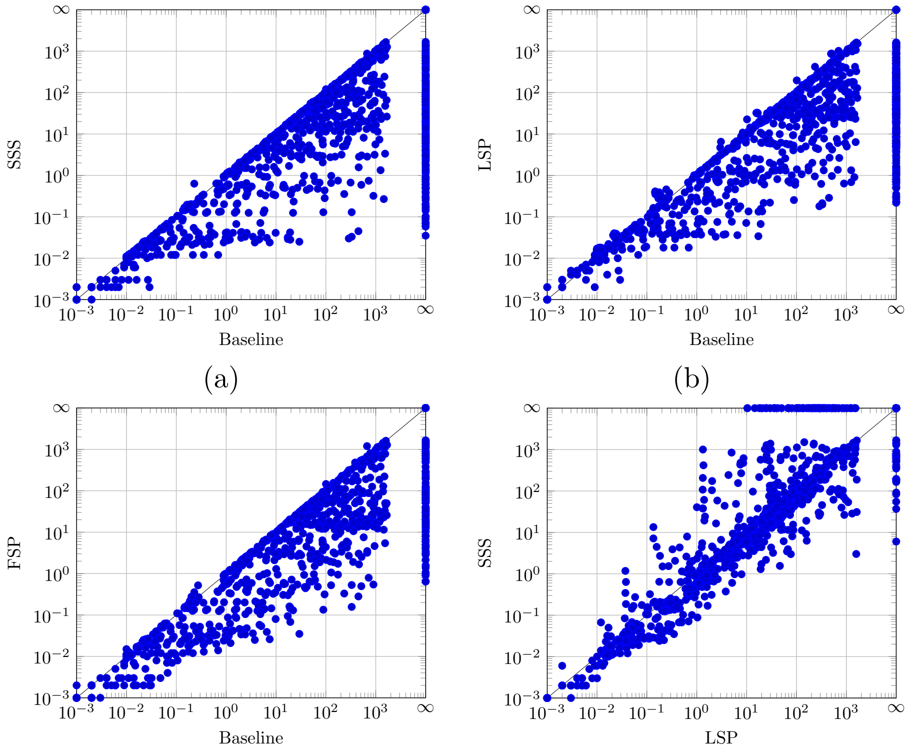
Comparing LSP vs. SSS in Figure 3.4 (d), we see that LSP has a noticeable advantage in larger instances, which also explains the slight edge in coverage. It is worth pointing out here that LSP can only be useful if the computed $\mathsf{up}^{L}_{\forall}$ indeed corresponds to the maximal follower response cost in S^*. In our experiments, this was the case in all instances where S^* could be fully constructed. This is, however, not guaranteed to happen in general; in other cases the other two pruning techniques may become more important. Regarding FSP, we remark that this technique is strongest in our transportation domains of type (ii), i.e., those with leader actions moving along the road map. As such actions never invalidate the follower's best response, this leads to perfect upper bounds and thus frequent follower-search pruning.

3.6.4 Greedy Stackelberg Planning

We noticed that the computational cost of the optimal follower search is critical in many cases, and thus ran experiments with a greedy variant of our algorithms. The only change is that we use a satisficing instead of an optimal planner at the follower level. This makes the follower-level search way more effective, at the price of no longer giving a guarantee on the correctness of the output (the Pareto frontier may be faulty as some leader states may have been assigned an overly high follower cost). Note that the curve induced by the greedy Pareto frontier solution can only be on or above the curve induced by the optimal solution.

Specifically, at the follower level we now run the commonly used baseline satisficing planner, Fast Downward's greedy best-first search (GBFS) with a dual queue for preferred operators, using the inadmissible heuristic h^{FF} [HN01]. To effectively prove follower problems unsolvable, as before we use Aidos' PDB heuristic.

The coverage results are included with the ones for optimal Stackelberg planning in Figure 3.2 and Figure 3.3. We observe a dramatic coverage increase in those domains



(a)

(b)

(c)

(d)

Figure 3.4: Per instance comparison of runtime for (a) no pruning (x) vs. SSS (y); (b) no pruning (x) vs. LSP (y); (c) no pruning (x) vs. FSP (y); (d) LSP (x) vs. SSS (y).

– Logistics and Visitall– where computing optimal follower responses is prohibitively costly. In the other domains, the difference to optimal Stackelberg planning is relatively small. For Pentest and Nomystery, the follower level is not the bottleneck of the overall search; for Sokoban, even optimal Stackelberg planning has close to maximal coverage anyway.

Switching from optimal to satisficing follower search in Nomystery increases coverage by about 10 percentage points. The influence of the follower level is marginal in this domain, and LSP pruning has a much higher impact (FSP + SSS and the baseline fall short of the other configurations). Similarly for Pentest, where the coverage gain from optimal to greedy Stackelberg planning is even less pronounced.

That difference is huge though in Logistics and Visitall. In both domains, the coverage of the best greedy configuration (FSP + SSS + LSP) starts at about 90% and decreases linearly in the number of leader actions. Again, only the baseline and FSP + SSS configurations are considerably weaker.

For the Visitall domain, we discovered that the sub-optimality of greedy follower-level planning positively influences the impact of LSP pruning: with a satisficing follower planner, it can happen that the upper-bound $\mathsf{up}^L_\forall$, though obtained from a very coarse approximation, is *smaller* than the follower costs computed by the satisficing planner. This is indeed often the case in Visitall, where our follower planner is prone to choose highly suboptimal routes across the map. To our leader-level search, it then appears that many of the follower responses encountered in search are worse than the already known upper bound, so that LSP pruning cuts off the search very early. Computational performance thrives on this, yet the returned Pareto frontiers are highly incomplete.

Importantly, Visitall is the only domain in which we observed this phenomenon. To back this up, we compared the quality of the greedy vs. the optimal solutions for all domains. We measured the size of the areas beneath the optimal and greedy Pareto frontier curves as a comparison. With the most competitive configuration (FSP + SSS + LSP), the areas under the greedy curves are on average about 10% larger than the areas under the optimal ones. An exception is the visitall domain where the area size is doubled.

3.7 Conclusion

Stackelberg planning is, we believe, an exciting new variant of planning that may be of high practical value in a large range of potential planning applications. First, it allows modeling defenses against attackers encoded as planning agents. Second, it subsumes a notion of robustness against sabotage and against worst-case perturbances by an environment. From an algorithmic point of view, it provides an interesting middle ground between full-scale game-theoretic planning and classical planning, generalizing the latter to a single adversarial exchange of action sequences. Our first results suggest that many of the successful algorithmic techniques from classical planning can be lifted to this setting. For example, we introduce Symbolic Leader Search utilizing the classical planning technique *Symbolic search* in the following chapter.

4

Faster Stackelberg Planning via Symbolic Search and Information Sharing

4.1 Motivation and Contributions

In the previous Chapter 3, we introduced the Stackelberg planning framework inspired by Stackelberg security games [Tam11; SJY^{+}14]. It models a single exchange of adversarial plan choice between two agents, *leader* and *follower*, acting in the same planning task. The leader's objective is to maximize the follower's plan cost. Solutions form a Pareto front containing pairs of leader/follower plans. This captures security-related scenarios like network security, where the leader must make it as hard as possible for an attacker to harm the system. It can also capture robustness-related scenarios, by assessing the impact of an adversarial event (constructed by the leader) on the cost of the follower's plan (the agent acting in the domain at hand).

In Section 3.3, we introduced a search algorithm in the space of leader actions, where each node corresponds to solving a follower task. Thus, one may need to solve exponentially many follower subtasks, each of which is a cost-optimal classical planning task.

We introduce Symbolic Leader Search (**SLS**), a new algorithm for solving Stackelberg planning tasks. **SLS** aims to effectively reuse as much information as possible between the different subtasks. We achieve this (1) by recognizing that most follower subtasks can be solved with bounded cost suboptimal planning techniques, which are often more efficient than optimal planning algorithms; and (2) by using a symbolic representation to share information across searches. Instead of considering the subtasks in isolation, **SLS** symbolically represents all possible follower subtasks that the leader can reach with a given cost. Not only does this avoid the explicit enumeration of all possible states in the leader state space, which can potentially provide an exponential advantage in terms of memory and time, but it also allows for sharing information, e.g., by reusing upper-bound functions for cost-bounded planning algorithms.

Besides this algorithmic contribution, we also introduce an extension of Stackelberg planning, using soft goals and net-benefit planning at the follower level. They generalize the follower's goal and naturally occur both in the security- and robustness-related scenarios. In the former, we can now model attackers that wish to inflict maximal damage (e.g., sum of users associated to compromised domains). In the latter, we can now model agents that maximize the achievable benefits subject to the damage inflicted by the adversarial event. Solving such more general Stackelberg planning tasks is, in principle, straightforward: we can use the well-known compilation from net-benefit to classical planning [KG09].

Our empirical evaluation shows that **SLS** outperforms our first approach for Stackelberg planning. The improvement is consistent, and is particularly pronounced in tasks with large leader action spaces and in net-benefit Stackelberg planning where follower subtasks are inherently harder to solve. We thus significantly extend the scope of Stackelberg planning tools.

4.2 Background

We introduced Stackelberg planning following the classical STRIPS framework in Section 3.2. In this section, we first provide the background on classical SAS$^+$ planning tasks and afterwards demonstrate how Stackelberg planning can be formalized in the SAS$^+$ framework.

4.2.1 Classical SAS$^+$ Planning and Symbolic Search

A SAS$^+$ classical planning task is a tuple $(\mathcal{V}, \mathcal{A}, \mathcal{I}, \mathcal{G})$ where $\mathcal{V}$ is a set of variables of finite domain [BN95]. A state is a value assignment to all variables in $\mathcal{V}$, and a partial state p is an assignment to a subset of variables $\mathcal{V}_p \subseteq \mathcal{V}$. Given a (partial) state s, and a set of variables $V \subseteq \mathcal{V}_s$, $s|_V$ denotes the projection of s onto V. Given two pairs of partial states s, t with disjoint variables, we define their union $s \cup t$ as a partial state over $\mathcal{V}_s \cup \mathcal{V}_t$ that agrees with both s and t. A partial state s satisfies another p (written $s \models p$) if $s[v] = p[v]$ for all $v \in \mathcal{V}_p \cup \mathcal{V}_s$. $\mathcal{I}$ is the initial state, and $\mathcal{G}$ is a partial state that represents the goal. $\mathcal{A}$ is a set of actions, where each action $a \in \mathcal{A}$ has a precondition, $\mathit{pre}(a)$, and an effect $\mathit{eff}(a)$, which are partial states. An action a is applicable in a state s iff $s \models \mathit{pre}(a)$. The resulting successor of applying a in s is $s[[a]] = s|_{\mathcal{V} \setminus \mathcal{V}_{\mathit{eff}(a)}} \cup \mathit{eff}(a)$.

A plan is a sequence of actions that go from $\mathcal{I}$ to any state s s.t. $s \models \mathcal{G}$. Moreover, each action has a cost $c(a) \in \mathbb{R}_0^+$, and the cost of a plan is the summed up cost of all its actions. A plan is optimal if it has a minimum cost.

Optimal planning is the problem of finding an optimal plan for any given planning task. Cost-bounded planning is the problem of, given a planning task and a cost bound B, returning a plan that has cost lower or equal to B or, if no such plan exists, returning "unsolvable".

Symbolic search is a well-known approach for exhaustive state space exploration in model-checking [McM93] and cost-optimal planning [EK09; TAK$^+$17]. In symbolic search, Binary Decision Diagrams (BDDs) [Bry86] are used to compactly represent sets of states as functions that map variable assignments to true or false, depending on whether such an assignment belongs to that set or not. We use the term V-BDD to denote a set of partial assignments over a specific subset of variables $V \subseteq \mathcal{V}$. BDDs offer a compact representation that often has an exponential gain in memory efficiency over listing all states in the set. Moreover, standard BDD operations can be used to operate on sets of states, e.g., the union of two sets of states corresponds to the disjunction of their BDDs ($S \vee S'$), and their intersection corresponds to the respective conjunction ($S \wedge S'$). The runtime of these operations depends on the BDD size, but not on the number of states in the set, translating the gain in memory to a gain in time.

4.2.2 Stackelberg Planning in SAS$^+$ Framework

Following the classical SAS$^+$ framework, a Stackelberg planning task is a tuple ($\mathcal{V}$, $\mathcal{A}^L$, $\mathcal{A}^F$, $\mathcal{I}$, $\mathcal{G}$). The set of actions is split into the leader actions $\mathcal{A}^L$ and the follower actions $\mathcal{A}^F$. The set of variables $\mathcal{V}$can thus be categorized into three subsets. Leader ($\mathcal{V}^L$) and follower ($\mathcal{V}^F$) variables are those that appear in the effect of a leader/follower action. The subtask variables ($\mathcal{V}^T \subseteq \mathcal{V}^L$) are those leader variables that appear in the goal or in the precondition of a follower action. Note that these subsets may overlap. We assume w.l.o.g. that $\mathcal{V}_\mathcal{G} \subseteq \mathcal{V}^F$.

Each sequence of leader actions π^L that is applicable on $\mathcal{I}$ corresponds to an assignment to $\mathcal{V}^T$, which is $\mathcal{I}[[\pi^L]]|_{\mathcal{V}^T}$. Given an assignment X to $\mathcal{V}^T$, one can construct a planning task $\Pi_X = (\mathcal{V}^F, A_X, \mathcal{I}_X, \mathcal{G})$ that corresponds exactly to the follower subtask. All such tasks are defined over the same set of variables, $\mathcal{V}^F$, and contain the subset of follower actions $A_X \subseteq \mathcal{A}^F$ such that preconditions over variables in $\mathcal{V}^T \setminus \mathcal{V}^F$ are satisfied by X. The initial state $\mathcal{I}_X$ is $X|_{\mathcal{V}^F} \cup \mathcal{I}|_{\mathcal{V}^F \setminus \mathcal{V}^T}$.

A pair (π^L, π^F) is a Stackelberg plan if π^L is a sequence of leader actions applicable on $\mathcal{I}$, and π^F is an optimal plan for the follower subtask $\Pi_{\mathcal{I}[[\pi^L]]}$. A Stackelberg plan (π_1^L, π_1^F) is dominated by another (π_2^L, π_2^F) if $c(\pi_1^L) \geq c(\pi_2^L) \wedge c(\pi_1^F) \leq c(\pi_2^F)$. The domination is strict if one of the inequalities is strict. The Stackelberg planning problem is computing a Pareto front, i.e., a set of non strictly-dominated Stackelberg plans which dominate all other Stackelberg plans.

Let Π^S be a Stackelberg planning task, and PF a Pareto front of Π^S. We define the set of cost entries of PF as $c(PF) = \{(c(\pi^L), c(\pi^F)) \mid (\pi^L, \pi^F) \in PF\}$. Note that, even though there are many possible Pareto fronts for a planning task, they all have the same set of cost entries. Moreover, it is sufficient to include a single plan (π^L, π^F) in the Pareto front for each cost entry. Therefore, each task has a unique Pareto front size defined as $|PF(\Pi^S)| = |c(|PF(\Pi^S)|)|$.

Stackelberg tasks are solved by performing a search in the space of leader actions, and solving the corresponding follower subtask at every node. In the preceding Chapter 3, we proposed to use iterative-deepening search (IDS) on the leader state space. This allows for a simple but important optimization, caching the plan of the parent and testing whether the same plan is a solution for any successor. This avoids many unnecessary calls to the follower subsolver, greatly speeding the search.

Another important enhancement is *upper-bound pruning.* Given a global upper bound on the follower cost, we can prune any leader state that has leader cost greater than any other state with a follower cost equal to the upper-bound. This upper bound can be determined whenever $\mathcal{V}^L \cap \mathcal{V}^F = \emptyset$.[1] In that case, all follower subtasks have the same initial state and, therefore, a follower task, Π^+, can be defined where all actions that can be disabled by the leader have been removed. As removing actions can only increase a task's solution cost, the follower cost of Π^+ is an upper bound on the follower cost of any other follower subtask.

4.3 Symbolic Leader Search

Algorithm 4.1: Symbolic Leader Search (**SLS**).

```
Input: Stackelberg Task $\Pi^S = (\mathcal{V}, \mathcal{A}^L, \mathcal{A}^F, \mathcal{I}, \mathcal{G})$
Output: Pareto front
$c^L \leftarrow 0$, $c^F \leftarrow -1$ ;
$F^+ \leftarrow$ GlobalUpperBound($\Pi^S$) ;
$Solved \leftarrow \emptyset$;
$ParetoFront \leftarrow \emptyset$;
while $c^F < F^+ \wedge c^L < \infty$ do
    $S^L \leftarrow$ UniformCostSearchLayer($c^L$) ;
    $S^T \leftarrow S^L|_{\mathcal{V}^T} \wedge \neg Solved$ ;
    while $S^T \neq \emptyset$ do
        $\Pi^F \leftarrow$ ChooseSubtask($S^T$) ;
        $plan, lb, ub \leftarrow$ Planner($\Pi^F, c^F, lb, ub$) ;
        if plan not found then
            $c^F \leftarrow \infty$ ;
            $S^T \leftarrow \emptyset$ ;
        else
            $c^F \leftarrow \max(c^F, cost(plan))$ ;
            $sol, ub \leftarrow$ RegressPlan($plan, ub$) ;
            $S^T \leftarrow S^T \wedge \neg sol$ ;
            $Solved \leftarrow Solved \vee sol$ ;
    AddIfNotDominated($ParetoFront, c^L, c^F$) ;
    $c^L \leftarrow nextL$;
return ParetoFront ;
```

Like our first implementation in Chapter 3, our main algorithm, symbolic leader search, **SLS** performs an exploration on the space of actions for the leader, using classical planning algorithms to solve the follower subtasks that result from leader plans. Thus, we can divide the overall algorithm into two parts: the search on the leader space and the follower subtasks. To solve the follower subtasks, previous work considered an optimal planner using explicit A*search with the LM-cut heuristic [HD09]. We also consider symbolic bidirectional blind search [TAK$^+$17], which, as we will discuss in the

[1] This is a natural property whenever the leader has higher-level actuators than the follower: e.g., changing a road network versus moving in it; changing a network configuration versus trying to break into it. In these cases, the effect of leader actions (e.g., block a road, update a server) cannot be undone by the follower.

following subsections, has a synergy with **SLS** due to employing backward search and a symbolic representation.

SLS focuses on how to reuse information among subtasks. Outside of the subsolver, **SLS** (1) exploits the follower cost of previous follower subtasks that have been optimally solved as a bound for subsequent calls to the follower solver; and (2) keeps track of the set of follower subtasks that are solved by some previously found plan. Inside the subsolver, three sources of information are exploited: a lower bound c^F on the follower cost on other subtasks with the same or lower leader cost, and two functions *lb* and *ub*. These functions allow for reusing information across follower subtasks and are discussed in detail in the next subsection.

SLS (see Algorithm 4.1) enumerates the leader search space layer by layer, using symbolic uniform-cost search on the space of leader actions over the set of leader variables. After exhausting a layer, the uniform cost search returns a set of leader states S^L that is reachable with leader cost c^L. These states are represented as a $\mathcal{V}^L$-BDD, avoiding their explicit enumeration, so that the algorithm can scale in tasks with large leader state spaces. Exploring the leader space layer by layer allows us to keep track of several global variables, namely c^L, c^F, the current Pareto front, and *Solved*. c^L is the current leader cost. As all layers of leader cost up to c^L have already been fully explored, the Pareto front up to that point has already been computed and can be reported up front, providing the user with partial results before the entire search is finished. Moreover, we also keep track of the current follower bound c^F, which is the highest optimal solution cost for any solved subtask with leader cost c^L or less. All subtasks with an optimal follower cost lower or equal to c^F do not contribute to the final Pareto front, so optimal solutions are not needed for them. Finally, we keep track of a set of solved subtasks, *Solved*, which are known to have an optimal cost lower or equal than the current c^F bound.

In each iteration, the first step is to obtain the set of candidate subtasks that could possibly add a new entry in the Pareto front, S^T, represented as a $\mathcal{V}^T$-BDD. This is done with standard BDD operations (see line 7), projecting S^L onto those leader variables that are relevant for the follower ($\mathcal{V}^T$), and subtracting all previously solved subtasks.

While there are follower subtasks in S^T that remain to be solved, one of them is chosen and solved by the follower solver. If the follower task is unsolvable, then we have found the last entry of the Pareto front (with $F = \infty$) and the algorithm ends. Otherwise, we update the value of c^F. The function RegressPlan returns the set of follower subtasks for which the same plan can be applied, as detailed in the next section. Different strategies could be applied, as long as it is guaranteed that the current subtask belongs to this set, so that no subtask is solved more than once. Once again, while there may be exponentially many such subtasks, we take advantage of a compact BDD representation to keep this computation tractable. All those subtasks are removed from S^T with a single BDD operation (line 17), possibly avoiding the enumeration of exponentially many leader states.

One important detail when using BDDs is the variable ordering, since the size of the BDD (and therefore the memory and running time of the algorithm) can be heavily affected by exponential factors under different variable orderings [Bry86; KH14]. We split the variables into three partitions ordered from top to bottom: (i) leader-only variables $\mathcal{V}^L \setminus \mathcal{V}^F$, (ii) common variables $\mathcal{V}^L \cup \mathcal{V}^F$, and (iii) follower-only variables $\mathcal{V}^F \setminus \mathcal{V}^L$. The intuition is that this simplifies the operations that project out these sets of variables (e.g., in line 7 of Algorithm 4.1). The order inside each partition is decided independently by a standard algorithm used to optimize BDD variable orderings in planning [KE11], only considering leader variables for partition (i), and only considering follower actions for partitions (ii) and (iii). This is important when the follower solver employs symbolic bidirectional search, as taking into account leader variables when optimizing the order of follower variables may be harmful for the performance of the follower subsolver.

4.3.1 Cost-Bounded Follower Solvers

A core idea of our approach is that it is not necessary to compute the optimal solution for every follower subtask to obtain the optimal Pareto front, as done previously. Instead, it suffices to compute the optimal solution for those subtasks that belong to the optimal Pareto front (i.e., at most one for each leader cost). For all other follower subtasks, it is enough to find any solution of lower or equal cost than any entry in the Pareto front with lower or equal leader cost. For this, one can use specialized bounded cost search algorithms [TSF$^+$12; Has13; SFv$^+$14; DH13] using the current c^F-bound. In practice, this is often much more efficient, even when the bound is the optimal follower cost (and therefore an optimal solution must be found anyway), since we can avoid proving optimality.

More precisely, our follower subtask solvers receive as input a classical planning task Π, and a cost bound B. The corresponding solution is a plan of cost B or less if one exists. Otherwise, it must return an optimal plan if one exists or "unsolvable" otherwise. Note that this is somewhere between cost-bounded and optimal planning. A straightforward approach is to use a cost-bounded planner and run an optimal planner in case it fails to find a solution.

To transform any search-based follower solver into a cost-bounded version, we consider two functions that map states of the follower task to numerical values: *lb* and *ub*:

- *lb* is an admissible estimate, i.e., it provides a lower bound on the goal distance from every state.
- *ub* is an upper bound on solution cost such that for all non-goal states s, $ub(s) \geq \min_{a \in \mathcal{A}^F} c(a) + ub(s[[a]])$. This property ensures that a plan can be reconstructed in polynomial time in the size of the task and the plan length if there are no 0-cost actions.[2]

We assume that *lb* and *ub* are represented symbolically, as a function from upper bounds to $\mathcal{V}^F$-BDDs. That is, for each possible return value, we keep a BDD that represents the set of follower states with this value. This is a common way to represent heuristics in symbolic search planning [KE11]. Figure 4.1 summarizes how these functions are obtained, as detailed in the next sections, and used.

[2] In tasks with 0-cost actions, *ub* also needs to account for the number of 0-cost actions but we omit such details for simplicity.

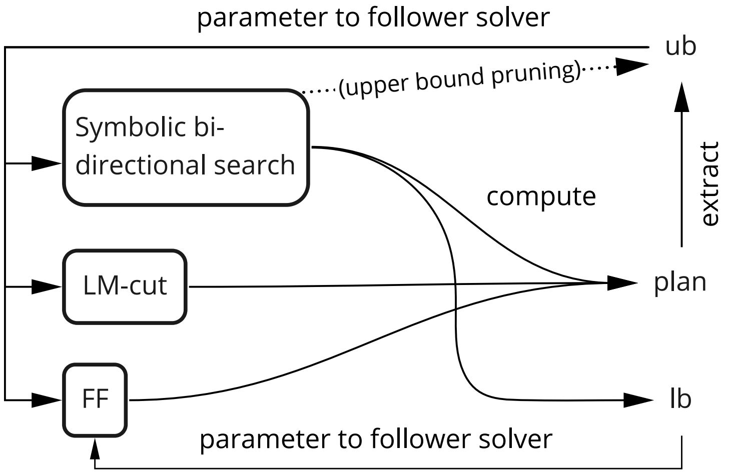

parameter to follower solver
ub
(upper bound pruning)
Symbolic bi-directional search
extract
compute
LM-cut
plan
FF
lb
parameter to follower solver

Figure 4.1: Computation and use of lower and upper bounds.

We modify all search algorithms to use *ub* as follows. During the search, any time a state s is generated, we check whether $g(s) + ub(s) \leq F$, where $g(s)$ is the cost of the path from $\mathcal{I}$ to s. If so, we stop the search and return a plan of cost $g(s) + ub(s)$ passing through s. As *ub* is represented symbolically, this is also possible in symbolic search.

To aggressively look for a solution below the cost bound B, we use a greedy best-first search with the FF heuristic [HN01] and a time limit of 1s, pruning any node whose g-value plus the lower bound *lb* is greater than B. Due to the time limit and the inadmissible heuristic, this is an incomplete configuration, so if it finishes without finding a solution within the bound, we run one of the optimal solvers on the same subtask.

Even though using *lb* is straightforward for A*-based solvers by taking the minimum among the heuristic and the value of *lb*, we do not use it here for practical reasons: extracting *lb* requires using the symbolic bidirectional search solver and none of our configurations use it with LM-cut. We do not use *lb* to bolster the symbolic search solver, as it performs blind search without using heuristics, and experimental analysis show that using heuristics is not always helpful in such setting [SGM20].

4.3.2 Transferring Bounds Across Subtasks

The similarity between follower subtasks can be directly exploited by sharing lower and upper bounds on solution cost among them. As they share a common set variables and a common goal, we can define dominance across follower subtasks in the following way:

Definition 1. *Let* Π_1 *and* Π_2 *be two planning tasks with a common set of variables. We say that* Π_1 *dominates* Π_2 *if* $h_1^*(s) \leq h_2^*(s)$ *for all states* s.

A sufficient criterion to identify that a task dominates another is to compare the set of actions.

Proposition 2. *Let* $\Pi_1 = \langle \mathcal{V}^F, A_1, I_1, G\rangle$ *and* $\Pi_2 = \langle \mathcal{V}^F, A_2, I_2, G\rangle$ *be two tasks with a common set of variables and goal, s.t.* $A_2 \subseteq A_1$*. Then,* Π_1 *dominates* Π_2*.*

Whenever dominance is established, we can use it to transfer bounds.

Proposition 3. *Let Π_1 and Π_2 be two tasks such that Π_1 dominates Π_2. Then any lower-bound function for Π_1 is a lower-bound for Π_2, and any upper-bound function for Π_2 is an upper-bound for Π_1.*

It is well known that symbolic backward search provides the exact goal distance on all states around the goal. This has been used in the past in order to compute admissible heuristics [TLB18]. Therefore, whenever symbolic bidirectional search is used as a solver, we use the perimeter created by the backward search as a lower and upper bound that can be transferred to other tasks.

This is mainly useful to obtain lower-bound functions for the FF solver, as follower subtasks are explored by increasing leader cost and, typically, leader actions tend to disable actions for the follower and not to enable them (since the objective of the leader is to increase follower cost). If that is the case, the initial task that is optimally solved at the beginning of the algorithm dominates all other tasks and therefore any symbolic backward search performed on it can be used as a lower-bound function on all subsequent subtasks.

Moreover, we can also transfer any backward search on Π^+ as an upper-bound function to all other subtasks. To boost this effect, whenever the upper-bound pruning optimization is enabled, we force symbolic bidirectional search to choose the backward direction for up to 30 seconds.

4.3.3 Plan Reuse Strategies

Inspired by a technique presented by Kolobov, Mausam, and Weld [KMW12] for probabilistic planning, we use the well-known notion of regression to *generalize* from individual follower plans. Generalization in our case pertains to two parts: (1) allowing for early termination of subsequent calls to the follower subsolver, and (2) removing entire follower subtasks from consideration.

Algorithm 4.2 depicts the function RegressPlan. It is called whenever a follower subtask has been solved in the leader search and a follower plan $\langle a_1, \dots, a_n \rangle$ have been found. (If the follower subtask was unsolvable, the leader search terminates anyway.) RegressPlan then identifies a condition C_i at every step $1 \leq i \leq n$ along the found plan such that for every state s, $s \models C_i$ implies that $\langle a_i, \dots, a_n \rangle$ is a follower plan for s. The conditions are computed by traversing the plan back to front, and computing in each step the regression of the respective action with the previously computed condition. In the regression, we must take into account all relevant variables $\mathcal{V}^F \cup \mathcal{V}^T$ to correctly compute these conditions. $\mathcal{V}^F$ alone identifies states within all follower tasks, while $\mathcal{V}^T$ provides the context, i.e., it constrains the follower subtasks to those in which the actions in the plan have not been disabled by the leader.

For the purpose of (1), the extracted conditions feed directly into the follower-cost upper bound *ub*. For every state that satisfies C_i, we have now found a plan with cost $c(\langle a_i, \dots, a_n \rangle)$. The *ub* function is updated with appropriate BDD operations (lines 3 and 7). When a new follower subtask $s^T \in S^T$ is chosen to be solved, *ub* is transformed into an upper-bound function for the corresponding follower subtask by computing

```
Algorithm 4.2: RegressPlan.
Input: Stackelberg Task ($\mathcal{V}$, $\mathcal{A}^L$, $\mathcal{A}^F$, $\mathcal{I}$, $\mathcal{G}$)
Input: Follower plan: $\langle a_1, \ldots, a_n \rangle$
Input: Previous upper bound function: $ub$
Output: Set of solved states and updated $ub$ function
$g \leftarrow 0$ ;
$C_{n+1} \leftarrow \mathcal{G}$ ;
$ub[g] \leftarrow ub[g] \vee BDD(C_{n+1})$ ;
for $i \in [n...1]$ do
    $C_i \leftarrow regress_{\mathcal{V}^F \cup \mathcal{V}^T}(C_{i+1}, a_i)$ ;
    $g \leftarrow g + c(a)$ ;
    $ub[g] \leftarrow ub[g] \vee BDD(C_i)$ ;
return $BDD(C_1|_{\mathcal{V}^T}), ub$ ;
```

$(ub[g] \wedge BDD(s^T|_{\mathcal{V}^T \setminus \mathcal{V}^F}))|_{\mathcal{V}^F}$ for each follower cost g. This restricts the upper bounds to those computed plans whose conditions projected onto the leader preconditions are still satisfied. In a final step, the resulting BDDs are projected onto $\mathcal{V}^F$ to obtain the $\mathcal{V}^F$-BDD that represents states of the follower subtask.

Regarding (2), consider the condition C_1 computed for the entire plan $\langle a_1, \ldots, a_n \rangle$. Since all variables of $\mathcal{V}^F \setminus \mathcal{V}^T$ in C_1 must necessarily be assigned to their respective values in $\mathcal{I}$, C_1 projected onto $\mathcal{V}^T$ hence identifies all subtasks with follower plan $\langle a_1, \ldots, a_n \rangle$. At this point, we know, however, that any new entry in the Pareto front must have an optimal follower plan cost higher than the cost of this plan. Hence, all subtasks identified by $C_1|_{\mathcal{V}^T}$ can safely be pruned. Note that $C_1|_{\mathcal{V}^T}$ always represents the subtask that was solved by the last call to the follower sub-solver. This property is required (and sufficient by itself) to guarantee the termination of the overall leader search.

4.4 Net-Benefit Stackelberg Planning

Our previous Stackelberg planning framework assumed that all follower goals must be achieved. However, for many applications, this is too restrictive. Consider, for example, the robustness of a road network in a logistics-like scenario. The follower's goal is to deliver packages from their respective pick-up location to different target locations. As soon as the leader blocks any pick-up or delivery location, the follower task becomes unsolvable. Consequently, all configurations where a package cannot be delivered have the same utility, regardless of how many packages still can. This limits the scope of the robustness analysis. It is more interesting to identify *how many* packages can still be delivered given the damage inflicted by the leader.

Similar in the pentesting scenario. Here, the follower tries to compromise a set of assets, i.e., computers holding sensitive information. Our previous experiments in Chapter 3 assumed the goal of accessing *all* assets. In that case, once one of the assets has been protected, the follower cost is infinite, even if other assets can be compromised. Vice versa, if the attacker's goal is to access *any* assets, then a defense measure that protects a single asset is regarded equally good as one that protects many more. In practice, assets hold different kinds of information that are valuated during risk assessment [TS05]. Therefore, it is desirable to quantify the leaked information by associating a utility to each piece of information. This is used in various DAG-based modeling approaches [KPS14], which are typically used on the local network level, or when quantifying the potential impact of compromised infrastructure in the Internet, e.g., name servers [RS05] or content delivery networks [SPR+17a].

Net-benefit Stackelberg planning allows the specification of soft goals, each being a partial state goal with an associated utility. The goal of the follower is to find a plan that maximizes utility minus cost. This is a natural way to model problems where the follower has more than a single goal.

Net-benefit Stackelberg planning tasks can be handled by the same algorithms as Stackelberg planning. We simply compile the utilities of soft goals into action cost, as done in classical planning [KG09]. Namely, for each soft goal we introduce an auxiliary action that achieves this fact[3] with a cost equal to the utility. These auxiliary actions remove a flag that is a precondition for all ordinary actions, so that once an auxiliary action achieves a soft goal, no ordinary actions are applicable.

4.5 Evaluation

We implemented our new **SLS** algorithm on top of the Stackelberg framework introduced in Chapter 3, built upon the Fast Downward planning system [Hel06]. We ran our experiments with the Downward Lab toolkit [SPS+17] on an Intel Xeon CPU E5-2650 v3, 2.30 GHz. For each task, we set a 30 minute time limit and a 4 GB memory limit, ignoring the translate and preprocessing phase which is equal for all configurations. Our source code, benchmarks, and results are publicly available [TSK+21a].

4.5.1 Benchmark Set

We evaluate our algorithms on three benchmark sets: Old, New, and Net. Old is the one established in Section 3.6. We include it to have a direct comparison with the same instances. It contains instances of classical planning IPC domains, but extended with n leader actions that disable preconditions of some follower actions. For each instance, we chose the versions ($n = 2, 4, 8, \dots, 512$) out of the 19 versions considered previously. Upper-bound pruning is applicable on all instances. All of them (with the exception of the Pentesting domain, cf.) involve the transportation of some objects to some locations. But the number of locations is small compared to the number of objects or goals, and thus succinct leader plans are often sufficient to render the follower's goal unsolvable. The left-hand side of Table 4.1 shows the maximum and average size of the Pareto front for each instance. Most solved instances in the Old set have very small Pareto front. Note that a Pareto front size of 1 means that the only Pareto-optimal choice for the leader is doing nothing. A size of 2 indicates that it has only one other option, which is typically to make some follower goal(s) unreachable with few actions.

New extends the previous benchmark set with instances better suited for the evaluation of Stackelberg planning algorithms. The domains in Table 4.1 with suffix

[3] We assume that soft goals are single facts. Otherwise an auxiliary fact representing the soft goal needs to be introduced as well.

Table 4.1: Maximum and average size of the Pareto frontier $|PF(\Pi^S)|$ on solved instances and coverage using LM-cut and symbolic bidirectional search subsolvers. We enable SLS's features one by one: reusing upper bounds (*ub*), upper-bound pruning (Π^+) and cost-bounded search with the FF heuristic (FF). We highlight the best configurations for each subsolver.

		$\|PF(\Pi^S)\|$		LMcut					Symbolic Bidirectional				
				IDS	**SLS**				IDS	**SLS**			
		avg	*max*	Π^+	—	+*ub*	$+\Pi^+$	+ FF	Π^+	—	+*ub*	$+\Pi^+$	+ FF
Old	Logistics-tcx (416)	1.85	3	**26**	**26**	**26**	**26**	**26**	**18**	**18**	**18**	**18**	**18**
	Mystery-tcx (218)	1.59	3	168	**172**	**172**	**172**	**172**	137	145	**151**	150	150
	Pentesting-tcx (213)	1.26	2	**201**	149	149	168	167	149	149	149	**168**	167
	Rovers-tcx (474)	1.86	3	**30**	**30**	**30**	**30**	**30**	50	69	68	69	**71**
	Sokoban-tcx (224)	1.92	2	**218**	**218**	**218**	**218**	**218**	190	**196**	195	**196**	**196**
	Tpp-tcx (132)	2.00	2	**4**	**4**	**4**	**4**	**4**	8	**9**	**9**	**9**	**9**
	Visitall-tcx (310)	2.32	7	34	31	**35**	34	34	32	35	38	**42**	41
	$\sum$ (1987)			**681**	630	634	652	651	584	621	628	**652**	**652**
New	Logistics-all (20)	4.60	6	3	3	**7**	**7**	**7**	**2**	**2**	**2**	**2**	**2**
	Logistics-tcx (120)	2.39	6	64	69	**83**	82	82	25	28	**29**	**29**	**29**
	Nomystery-all (20)	3.33	5	8	10	10	10	**11**	3	13	**14**	**14**	**14**
	Nomystery-tcx (108)	2.44	7	74	**76**	74	74	75	70	**93**	**93**	**93**	92
	Pentesting-all (26)	2.00	2	**26**	**26**	**26**	**26**	**26**	2	**25**	24	24	**25**
	Pentesting-tcx (259)	1.68	4	**259**	257	257	258	252	214	**257**	256	**257**	**257**
	Rovers-all (10)	2.11	3	**4**	**4**	**4**	**4**	**4**	5	**9**	8	8	**9**
	Rovers-tcx (92)	1.79	4	**36**	**36**	**36**	**36**	**36**	67	**77**	**77**	**77**	**77**
	Tpp-all (20)	3.88	5	**8**	**8**	**8**	**8**	**8**	7	**16**	**16**	**16**	**16**
	Tpp-tcx (110)	1.88	7	**63**	60	61	62	62	91	98	98	**102**	**102**
	Transport-all (20)	6.00	7	0	8	**10**	**10**	**10**	3	**12**	**12**	**12**	**12**
	Transport-tcx (120)	4.00	17	40	54	**58**	54	54	72	88	**90**	87	87
	Visitall-all (20)	3.00	3	14	14	**17**	**17**	**17**	8	**16**	**16**	**16**	**16**
	Visitall-tcx (114)	2.96	7	82	83	**92**	**92**	**92**	63	85	**88**	**88**	**88**
	$\sum$ (1059)			681	708	**743**	740	736	632	819	823	825	**826**
Net	Logistics-all (20)	9.50	12	0	0	4	5	**7**	0	1	**2**	**2**	**2**
	Logistics-tcx (120)	3.29	16	56	62	74	71	**76**	20	25	26	**27**	**27**
	Nomystery-all (20)	4.67	5	0	2	5	**6**	5	1	6	6	**11**	10
	Nomystery-tcx (108)	3.07	13	**61**	60	**61**	60	60	67	80	81	**86**	85
	Pentesting-all (27)	5.23	8	6	6	6	12	**13**	0	6	6	**9**	**9**
	Pentesting-tcx (263)	3.10	13	210	208	214	**238**	236	184	208	207	**215**	214
	Rovers-all (10)	7.25	9	**4**	**4**	**4**	**4**	**4**	3	7	**8**	**8**	**8**
	Rovers-tcx (92)	3.02	9	33	34	34	**36**	**36**	51	75	75	**79**	**79**
	Tpp-all (20)	4.38	7	**8**	7	7	**8**	**8**	6	10	11	**14**	**14**
	Tpp-tcx (110)	2.04	13	**53**	47	49	47	47	89	90	92	**102**	**102**
	Transport-all (20)	11.00	18	0	1	2	**6**	**6**	0	1	2	**6**	**6**
	Transport-tcx (120)	5.07	34	38	47	50	**52**	**52**	66	74	80	**84**	**84**
	Visitall-all (20)	5.14	7	7	4	6	**9**	**9**	6	6	4	**9**	7
	Visitall-tcx (114)	3.88	13	50	54	58	59	**60**	47	65	71	**74**	73
	$\sum$ (1064)			526	536	574	613	**619**	540	654	671	**726**	720

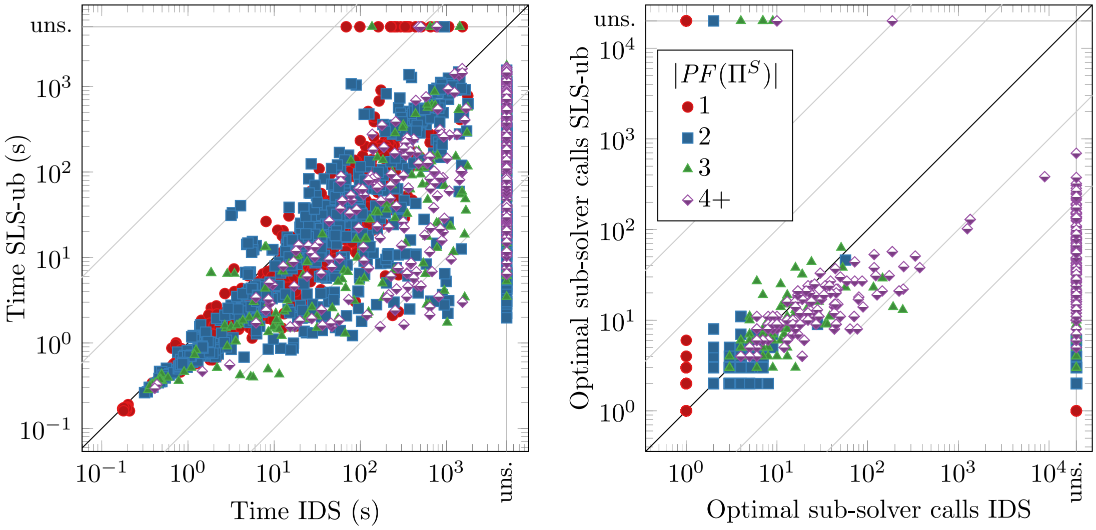


Figure 4.2: Total time and optimal follower searches for IDS and SLS-ub with symbolic bidirectional search.

"tcx" are the benchmark sets with scaled number of n leader actions. The suffix "all" represent the benchmark sets with all possible leader actions. We base our new instance set on the same domains, but increase the number of locations relative to the number of objects and goals. To do this, we generated instances of those domains with a size chosen s.t. the baseline planner can solve the instance in ≈10-60 seconds. Then, we scaled the number of locations. This provides the follower with more paths to the goal, and the leader with more options to increase the follower's cost, leading to larger Pareto fronts. We also replaced Mystery by the more modern No-mystery domain, and Sokoban (which does not admit a high number of locations without making the task significantly easier) with Transport. Compared to other domains, Transport features a more realistic road map, where locations are assigned coordinates in a plane and the action cost to travel between locations is their straight-line distance. This leads to larger Pareto fronts than any other domain.

NET is a set of net-benefit instances, discussed below.

4.5.2 Evaluation of SLS

Table 4.1 shows overall coverage. Of course, the choice of the underlying classical planner depends on the domain, e.g., LM-cut works best on Logistics and Sokoban, whereas symbolic bidirectional search is superior on Rovers, Transport and TPP. Our new algorithm, **SLS**, clearly outperforms the baseline, IDS from Chapter 3, with both solvers. The only exception is the Pentesting domain where the symbolic representation suffers due to the huge number of variables. The time spent at the leader search level is a low percentage of the total time, indicating that the main bottleneck is the time spent by the follower subsolvers. This means that the advantage of **SLS** mainly comes from two factors: (1) the number of calls to the follower subsolver, and (2) the speedup provided by sharing the bound functions.

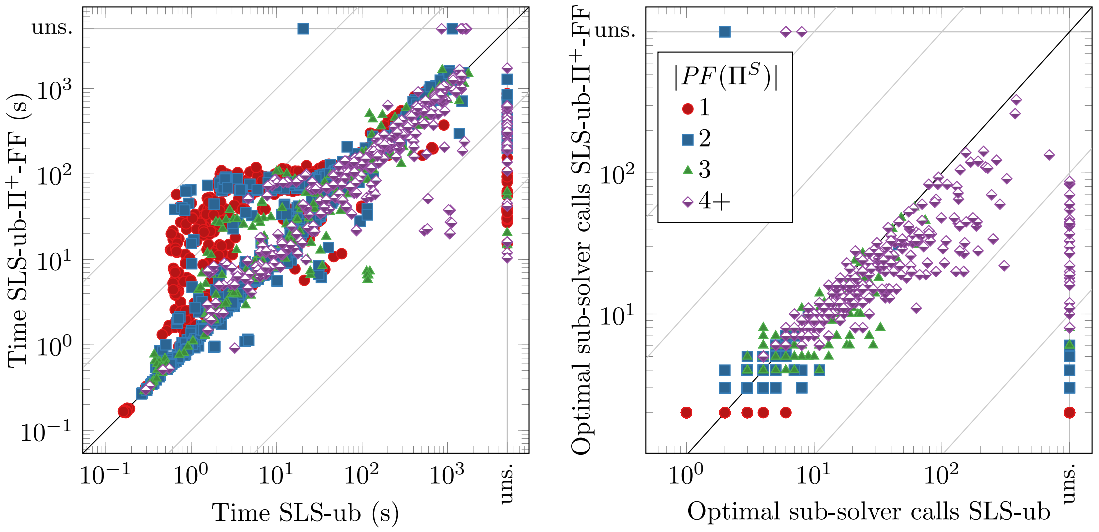


Figure 4.3: Total time and optimal follower searches of SLS-ub and SLS-ub-Π^+-FF with symbolic bidirectional search.

Our ablation analysis, enabling an optimization at a time, provides insights on what the main reasons for the advantage of **SLS** are. The main difference between IDS and the basic version of **SLS** is that the latter explores the leader space with symbolic search layer by layer, reusing all previously found plans, instead of only the parent's plan. Hence the performance difference between both is mainly due to the number of calls to the subsolver. Moreover, using the *ub* function is almost always beneficial, speeding up each call to the follower subsolver by allowing an early termination.

Figure 4.2 shows a comparison of the baseline IDS, and **SLS**-ub in terms of runtime and number of subsolver calls. The plot displays a clear trend regarding the planner performance with respect to the Pareto front size. Whenever, the Pareto front size has only 1 or 2 entries, few follower sub-searches are needed by both algorithms. In those tasks, the overall running time is only determined by how efficient the follower subsolver is and there is little room of improvement for reducing the number of calls to the subsolver or reusing information among different calls. Tasks with a Pareto front of 3 or more, however, require a much higher number of subsolver searches. By better reusing information among them, **SLS**-ub significantly reduces the number of searches, and the total time often by more than an order of magnitude.

Figure 4.3 shows in detail the effect of enabling upper-bound pruning (Π^+) and cost-bounded FF search. Both have up to 30s of pre-processing in the form of symbolic backward search to obtain upper (ub) and lower (lb) bounds. However, as the results of Table 4.1 indicate, this often pays off in terms of coverage and it is also slightly beneficial in runtime on instances where **SLS**-ub uses more than 100s.

4.5.3 Evaluation of Net-benefit Planning

To analyze the gap in difficulty between standard and net-benefit Stackelberg planning, we introduce the Net instance set. The instances are the same as in New, but consider

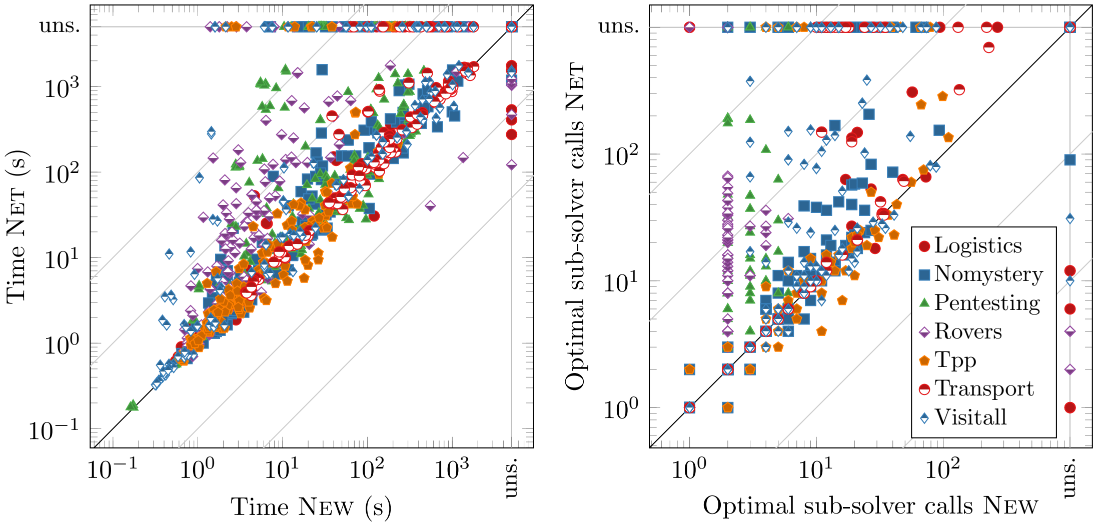


Figure 4.4: Comparison of New and Net instances when using SLS-ub with symbolic bidirectional search.

each goal fact to be an individual soft goal with a utility of 10000. This is significantly higher than the cost of any leader plan in these domains, so that the follower maximizes the number of achieved goals, while reducing its cost. Rather than modifying the PDDL encoding, we implemented the soft-goal compilation as a final step in Fast Downward's translator [Hel09]. Therefore, NET and NEW instances use the same variable representation so this constitutes a very direct evaluation of the impact on having soft goals.

Figure 4.4 compares **SLS**-ub on both instance sets, and other configurations have very similar results in this regard. The results show that net-benefit instances are often much harder, though the impact depends on the domain. Higher impact can be observed in domains like Rovers or Pentesting, where the average Pareto front size increased the most (see Table 4.1). Many NEW instances of these domains have tiny Pareto frontiers and require very few follower sub-searches. However, this is no longer true in their net-benefit counterparts. Coverage results show that **SLS** is particularly good on this set beating IDS even on the Pentesting domain, due to the higher proportion of hard tasks where many follower subtasks must be solved.

4.6 Conclusion

We introduced **SLS**, an algorithm for solving Stackelberg planning tasks. It exploits symbolic leader search and cost-bounded follower search to share information between subtasks. **SLS** thus consistently outperforms our previous approach, in particular when the leader action space is large or when we consider soft goals.

Part II

Applying Stackelberg Planning to Local Network Applications

Applying Stackelberg planning based algorithms for solving local network simulated penetration testing and mitigation analysis tasks

5 Playing CATSS and MAUS: a Theory of Privilege Escalations and Applications to Simulated Pentesting

5.1 Motivation and Contributions

80% of companies report unpatched vulnerabilities in their network [edg]. High-profile attacks like the WannaCry ransomware or the Equifax breach cause significant damage by exploiting month-old vulnerabilities. Given an estimated lack of 3M IT-Security experts [ISC20] worldwide and more and more legislation that mandates regular security assessments, e.g., the GDPR, we conclude that there is a dire need for automation. Indeed, security assessment is a thriving market [Mar].

A number of techniques and formalisms have been proposed for threat analysis, among those graphical models, e.g., attack graphs [SHJ^{+}02] and attack trees [WAF^{+}19], but also methods from model checking [RA00b], logic programming [OGA05] or planning [SSSH^{+}19]. Large networks tend to have vulnerable software, not necessarily because of a lack of know-how, but for compatibility or compliance reasons, or because usability was given preference over security. The security-conscious administrator tries to isolate the vulnerable hosts from valuable assets. Threat modeling techniques are thus focused on the probability of reaching valuable assets, typically by hopping from one vulnerable host to the next.

But sophisticated attackers ‘live off the land’; they obtain new privileges by running local exploits and seek passwords and cryptographic data that gives them access to other machines. As of yet, only a few of these models consider privileges on the same machine, and those who do consider only a few OS-level users in handpicked examples. To our knowledge, there is no theory for how privilege escalation works and how it can

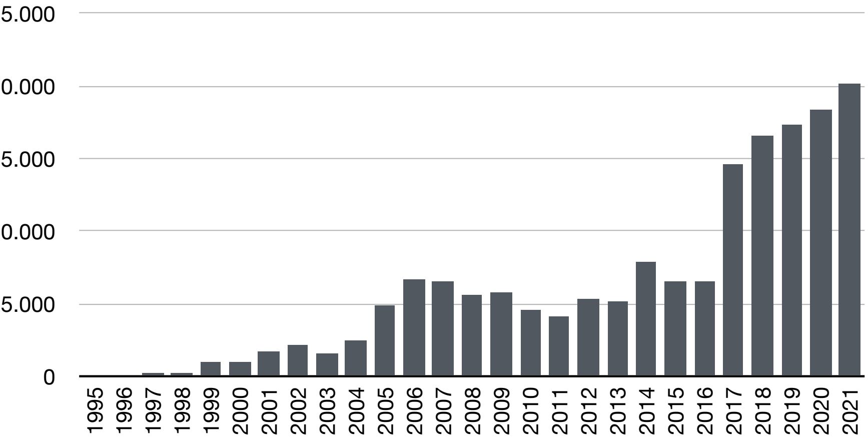


Figure 5.1: Number of new entries per year in the National Vulnerability Database since 1995.

be modeled effectively. All of the aforementioned threat analysis techniques stand to benefit from such a model, but a major obstacle is the data acquisition.

In practice, these models are generated by scanning for a rapidly growing number of known vulnerabilities (see Figure 5.1). It is hence not enough to propose a model, it should also be possible to generate models from the information available in public databases and local scans. Our theory points to the need for a just slightly more fine-grained classification of vulnerabilities in public databases. Adding just one bit of information, the consequences of a successful exploit can be modeled precisely in two categories, and reasonably approximated in the third.

Contribution In this chapter, we define a theory of privilege escalation and show its applications in a novel network scanning / risk analysis tool called CATSS/MAUS.

1. We formally define a theory of privilege escalation and provide a sound under approximation in terms of three operators in Section 5.3.

2. We discuss the relation of these operators to the existing CVSS standard, which defines the meta data in publicly available vulnerability databases in Section 5.4. Our theory exposes semantic ambiguities, a lack of granularity and information loss in the effects. By supplanting an existing boolean field with an assignment to one of the three operators, we can overcome these limitations.

3. Using Machine Learning, we demonstrate that this classification can be computed on the fly and with high enough accuracy to power new analyses. We evaluate the cost of mitigating the National Vulnerability Database in a coordinated effort.

4. We derive a Stackelberg planning model and implement a full-fledged vulnerability scanning and analysis toolchain called CATSS/MAUS in Section 5.6 and Section 5.7 based on the algorithms presented in Part I.

5. We engineered a methodology for setting up reproducible experiments in simulated pentesting (Section 5.9). We use it to evaluate our theory and implementation in

terms of accuracy and performance and allow others to scrutinize these results.

5.2 Related Work

A number of techniques for risk analysis in networks have been proposed, including graphical models, e.g., attack graphs [SHJ+02] and attack trees [WAF+19], model checking [RA00b], logic programming [OGA05] and planning [SSSH+19]. See [SSS14] for a survey about the use of attack graphs in security and [Hof15] for a perspective of simulated pentesting from the planning community. The focus of this work is the acquisition of a model, as opposed to the analysis technique. For CATSS/MAUS, we employ Stackelberg planning from Part I, but our model aims to be compatible with all techniques we have mentioned. We cover related work on the modeling aspect in the following.

The earliest methods for the generation of attack graphs from networks only cover connectivity [RA00b] or contain only a few hardcoded attacks that classify as privilege escalation [SHJ+02]. Some works discuss Unix users in theoretical examples [PS98b; XJS08], but neither discuss the relation between different hierarchies, nor how they can be used for vulnerability analysis in an automated fashion. MulVal [OGA05], perhaps the most elaborate and widely used tool for network attack graph generation, considers only OS-level users and uses a boolean to mark a vulnerability as a privilege escalation. Version 1.1 sets this boolean whenever a vulnerability has integrity impact.[1] Jing et al. [JYD+17] refine the effects of vulnerabilities by parsing CVSS metadata for keywords, but only consider remote attacks and three access levels, ‘none’, ‘normal‘ and ‘root’.

Bugliesi et al. [BCF+12] perform model checking on an RBAC implementation (grsecurity) by transforming a policy into a labeled transition system. This entails a more detailed user hierarchy, but, again, only for the operating system. By contrast, our model considers the relation *between* such hierarchies. This mechanism could, however, be integrated into our current prototype as a means to support grsecurity (see Section 5.7).

The closest related tool to our network scanner CATSS is the commercial scanner Nessus [Ten], which also allows for authorized scans. Other vulnerability scanners like OpenVAS [Gre] and NMAP [Lyo] or exploit frameworks like Metasploit [Met] provide a more limited view of hosts, but can operate without full access to the hosts they scan.

5.3 A Theory of Local and Remote Privilege Escalation

We present a theory that allows symbolic reasoning about privilege escalation attacks. It abstracts away from the software itself and focuses on the relation between *categories* of software in terms of ‘who controls whom’.

5.3.1 Access Control Mechanisms

We understand a privilege escalation attack as an attack that gives a malicious entity privileges it is not intended to have. The restriction of these privileges is governed by

[1] According to the source code.

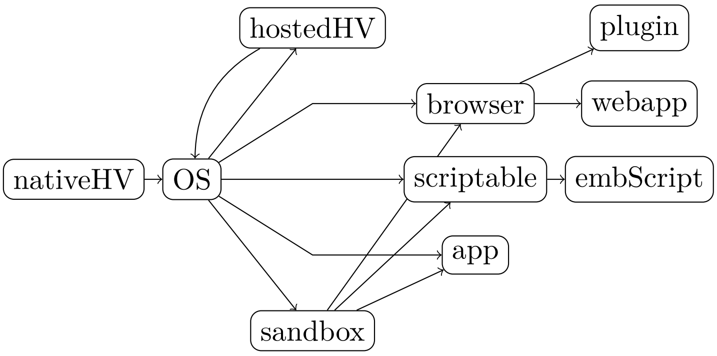


Figure 5.2: Simple software configuration graph.

lattice	popular software
$C_{nativeHV}$	Xen, Oracle VM Server, MS Hyper-V
C_{OS}	Linux, MacOS, Windows
$C_{sandbox}$	App-Sandbox, SELinux, mbox, Firejail, jail
C_{app}	(Apps from AppStore)
$C_{browser}$	Firefox, Chrome, Internet Explorer
C_{webapp}	Google Mail, Gitlab
$C_{scriptable}$	Microsoft Word, Vim
$C_{embScript}$	DOCX with macros
C_{plugin}	Adblocker etc.
$C_{hostedHV}$	VMWare Workstation, VirtualBox, QEMU

Figure 5.3: Example categorisation.

an *access control mechanism* (ACM), which is implemented by software or hardware. A privilege escalation is thus a fault in the implementation of the access control mechanism.

We model the ACM as a partially ordered set $(L, \leq)$ with a unique upper-bound $\top$. We call the elements of L privileges and L the *security lattice*.[2] For an operating system (OS) with discretionary access control, L is the set of users (subjects) and $s_1 \leq s_2$ if s_2 can access a superset of the objects s_1 can access.

5.3.1.1 Relation Between ACMs

A key observation is that ACMs are nested. An operating system restricts access to files and hardware. It executes programs that are themselves ACMs. A web browser, e.g., restricts access to cookies and other resources depending on the origin of the website it runs. The operating system itself is controlled by native or hosted hypervisor. To automatically compute the potential effects of a vulnerability, we assign software to categories and formalize the relation between these categories in a directed graph called the *software categorization graph.* In this graph, nodes are categories and two categories connect if software in the former is controlled by an access control mechanism in the latter (see Figure 5.2 and Figure 5.3). This categorization and graph is a matter of manual effort, however, in comparison to formalizing the effect of each vulnerability, categorizing all software is much easier. Let (C, E) be a fixed software categorization graph, where the nodes $C \subset 2^{\mathrm{S}}$ are categories (subsets) of software. Software can be identified, e.g., by their Common Platform Enumeration [CCK+11]. We write $s \xrightarrow{cat} s'$ if $s \in C$, $s' \in C'$ and $(C, C') \in E$.

The security lattices a software ought to implement is configuration specific and may change over time. We hence describe a system configuration as a set of tuples Sys. Each element of this set $acm \in \mathrm{Sys}$ is of form $(s, L, \leq, p)$ with s set up to implement $(L, \leq)$ and p the privileges with which this instance of s runs. Consider a database system: it implements an access control mechanism with a configuration-dependent security lattice. However, it also runs as a daemon with the privileges of a user in the operating system. These privileges are part of operating system's security lattice.

A vulnerability can therefore have a different effect depending on the configuration of a system. This includes the configuration of the vulnerable software, but also the access control mechanism it runs in. A vulnerability that allows arbitrary code execution effectively affords the adversary the privileges $\mathcal{P}$ of the vulnerable s itself. These privileges are part of the context in which s runs, more precisely, they are an element of the security lattice $(L, \leq)$ implemented by the software that runs s, e.g., the operating system.

Definition 2 (Sound system configuration)**.** *A system configuration (or system)* $\mathrm{Sys} \subset \mathrm{S} \times \mathcal{L} \times (\mathcal{L} \times \mathcal{L}) \times \mathcal{P}$ *is sound w.r.t. a software categorization graph* $(C, E) \subset \mathrm{S} \times (\mathrm{S} \times \mathrm{S})$ *if the graph* $(\mathrm{Sys}, \rightarrow)$ *with* $(s, L, \leq, p) \rightarrow (s', L', \leq', p')$ *if* $p \in L'$ *is a tree and* $(s, L, \leq, p) \rightarrow (s', L', \leq', p')$ *implies* $s \xrightarrow{cat} s'$*.*

[2] A lattice in the mathematical sense also has a lower bound, but this bound is not used in this work.

5.3.2 Local Privilege Escalations

We categorize vulnerabilities into three classes based on their potential effect and associate each with an operator $V_{\text{precise}} \subseteq \mathrm{S} \times \{\bot, \nearrow_p^{p'}, \Uparrow\}$. An exploit of vulnerability *cve* in s may cause

1. **no escalation** ($\bot$): the attacker's privileges remain the same,
2. **local escalation** ($\nearrow_p^{p'}$): the attacker obtains privileges p' *within* the ACM implemented by s that were not available to the attacker before,
3. **hyper escalation** ($\Uparrow$): the attacker obtains the privileges that s itself enjoys in the ACM that controls it.

We claim that this categorization is exhaustive: If a vulnerability affords the attacker new privileges that are not within the same ACM (and is thus not covered by 'no escalation' or 'local escalation'), then the new privileges must be those of the ACM itself — any other would be considered a vulnerability of the software implementing the parent ACM. We can assume this categorization to be exclusive if we require vulnerabilities that permit two types of escalation to be filed as two separate vulnerabilities.

Let $V, \mathrm{Sys}, p \vdash p'$ denote the statement that, given a system Sys and vulnerabilities V, any attacker starting with privilege p can obtain privilege p'. We define the following deduction system. The first two rules provide initial privilege and all privileges below those that can be obtained.

$$\text{I}\ \frac{}{V, \mathrm{Sys}, p \vdash p} \qquad \text{M}\ \frac{(s, L, \leq, \cdot) \in \mathrm{Sys} \qquad p' \leq p}{V, \mathrm{Sys}, p \vdash p'}$$

From non-escalations, the attacker obtains no privileges. From local escalations, the attacker gains the target privileges if the necessary privileges to mount the attack are obtained.

$$\text{L}\ \frac{(s, \nearrow_{p_{\text{from}}}^{p_{\text{to}}}) \in V \quad (s, L, \leq, \cdot) \in \mathrm{Sys} \quad V, \mathrm{Sys}, p \vdash p_{\text{from}}}{V, \mathrm{Sys}, p \vdash p_{\text{to}}}$$

Finally, for hyper escalations, the privileges gained are those under which the exploitable access control mechanism itself was running.

$$\text{H}\ \frac{(s, \Uparrow) \in V \qquad (s, L, \leq, p_{\text{up}}) \in \mathrm{Sys} \qquad V, \mathrm{Sys}, p \vdash p' \qquad p' \in L}{V, \mathrm{Sys}, p \vdash p_{\text{up}}}$$

These three rules are enough to describe local privilege escalations and contain enough structure to describe their effect. Note that, in particular, the effect of a hyper escalation is fully described by the $\Uparrow$ operator and the system configuration. Unfortunately, this is not the case for local escalations, which require a description of source and target privileges. These may be system dependent, but a vulnerability database like the NVD has to describe these effects in a language that is system-independent, but can be instantiated on a system. Moreover, this language would need to apply to practically all software. We doubt that such a lingua franca can be practical, and instead over-approximate this operation by defaulting to the top element of the security lattice and requiring an arbitrary permission. Let $V_{\text{coarse}} \subseteq \mathrm{S} \times \{\bot, \uparrow, \Uparrow\}$ and

$$\text{L2}\ \frac{(s,\uparrow) \in V \qquad (s, L, \leq, \cdot) \in \mathrm{Sys} \qquad V, \mathrm{Sys}, p \vdash p' \qquad p' \in L}{V, \mathrm{Sys}, p \vdash \top_L}$$

Corollary 1. *The inference system I,M,H,L2 is an over-approximation of the inference system I,M,H,L, i.e., each judgment. $V, \mathrm{Sys}, p \vdash p'$ that is derivable in the latter is also derivable in the former. This is because any judgment that can be derived from L, can be derived from L2 followed by M.*

5.3.3 Remote Privilege Escalation

$$\dfrac{\dfrac{\dfrac{\dfrac{V, W, p_{U_1} \vdash p_{U_1} \qquad (\text{gitlab}, \uparrow) \in V \qquad (\text{gitlab}, L_{\text{gitlab}}, \leq, \cdot) \in W \qquad p_{U_1} \in L_{\text{gitlab}}}{V, W, p_{U_1} \vdash \top_{L_{\text{gitlab}}}} \qquad p_{U_2} \leq \top_{L_{\text{gitlab}}}}{V, W, p_{U_1} \vdash p_{U_2}} \qquad W \in \mathrm{N}}{V, \mathrm{N}, \xrightarrow{cred}, p_{U_1} \vdash p_{U_2}} \qquad (W, p_{U_2}) \xrightarrow{cred} (C, p_{\text{origin:W}})}{V, \mathrm{N}, \xrightarrow{cred}, p_{U_1} \vdash (C, p_{\text{origin:W}})}$$

Figure 5.4: Proof tree for Example 1.

We now move beyond local privilege escalation and to the case of various systems in a network $\mathrm{N} = \{\mathrm{Sys}_1, \ldots, \mathrm{Sys}_n\}$ that may share credentials to each other. A credential is a piece of information that can be used by the adversary to obtain privileges on a system, e.g., a stored password or a cryptographic key. We model the credentials within a network at some time as a relation $\xrightarrow{cred} \subset (\mathrm{N}, \mathcal{P})^2$ between pairs of system configurations (or systems) and privileges s.t. $(acm_1, p_1) \xrightarrow{cred} (acm_2, p_2)$ iff the privileges $\mathcal{P}_1$ on system acm_1 afford the attacker the access to the necessary credentials to obtain privileges $\mathcal{P}_2$ on system acm_2. This relation needs to be sound, i.e., for each (acm_i, p_i), $i = 1, 2$ in the relation, there must indeed be an access control mechanism $(s, L, \leq, p) \in acm_i$ such that $p_i \in L$, and acm_i must itself be sound. We assume each system and security lattice to have distinct privileges (i.e., privileges are system- and software specific) and write $p \xrightarrow{cred} p'$ whenever the access control mechanism is not important. Now remote privilege escalation is modeled with the smallest possible relation $\vdash$ s.t.

$$\frac{V, \mathrm{Sys}, p \vdash p' \qquad \mathrm{Sys} \in \mathrm{N}}{V, \mathrm{N}, \xrightarrow{cred}, p \vdash p'} \qquad \frac{V, \mathrm{N}, \xrightarrow{cred}, p \vdash p' \qquad p' \xrightarrow{cred} p''}{V, \mathrm{N}, \xrightarrow{cred}, p \vdash p''}$$

The following example showcases how an elaborate vulnerability can be described in this theory.

Example 1. *Consider a stored XSS vulnerability cve $\in V$ for the popular project management platform Gitlab hosted on some web server W. In a nutshell, a stored XSS vulnerability means that an attacker can store JavaScript code on W so that it is later delivered to other users visiting W and executed in their browser's context. Thus, cve would be tagged as a local escalation, as it is executed with the privileges of the* *other user: cve* $= (\text{gitlab}, \uparrow)$*. Let* p_{U_2} *denote some privileges sufficient to deliver active content to some other user* U_2*.*[3] *Given that* $(\text{gitlab}, L_{\text{gitlab}}, \leq, \cdot) \in W$*, (as the software is installed) and* $p_{U_2} \leq \top_{L_{\text{gitlab}}}$*, an attacker that has any privilege* $p_{U_1} \in L_{\text{gitlab}}$ *can obtain this privilege.*

MAUS can detect that, on a client machine C*, a web browser stores the credentials of* U_2 *on* W*. This is represented by the relation* $(C, p_{\text{origin:W}}) \xrightarrow{cred} (W, p_{U_2})$ *for* $p_{\text{origin:W}}$ *the privileges of active content that is executed within that browser.*[4]*. Observe, however, that this relation is mutual: the active content controlled by* U_2 *is executed in the web browser:* $(W, p_{U_2}) \xrightarrow{cred} (C, p_{\text{origin:W}})$*. This can be detected, e.g., by analyzing the browser's password store, history or cookies. In a network* $N \supseteq \{W, C\}$*, it is thus possible for an attacker that has privileges* p_{U_1}*, to obtain privileges* $p_{\text{origin:W}}$*. See Figure 5.4 for a proof tree formalizing this argument.*

5.3.3.1 Relationship to Access Control Models

There are two kinds of access control models:

- ACL-based: Access of subjects' access to object is determined by a list. How this list is edited, and by whom, is a question of the system in use. The most prevalent systems are DAC, MAC, RBAC.
- Capability-based: unforgeable objects called capabilities, who can be passed from subject to subject, determine access to object.

If, e.g., a process runs with a set of privileges of an access control context that implements capability-based access control, then this set of privileges is determined by the capabilities that this process has at the time it is running. While our model is static, and thus only captures a moment, it can be extended to the dynamic case by modeling possible transitions with each respective security lattice.

5.4 Compatibility to Open Standards

To limit the exposure to known attack vectors, organizations monitor public vulnerability databases for threats that affect software they have deployed. The largest such database is the National Vulnerability Database (NVD), operated by the US government. It employs a series of public standards, called the Security Content Automation Protocol (SCAP) to enable automated vulnerability management and policy compliance evaluation. Among other standards, SCAP includes:

- CVE (Common Vulnerabilities and Exposure) to provide unique numerical identifiers for vulnerabilities,

[3] A fine-grained lattice can distinguish between the full set of privileges of a registered user and the ability to deliver active content after login. In practice, this is simplified due to the approximative operator ↑.

[4] Most browsers implement a same-origin policy, thus $p_{\text{origin:W}}$ is different for each website W. There is also access control between windows/tabs and the origin contains more than just the domain name; we gloss over these subtleties here. An accurate security lattice can be derived from the comprehensive formalization of most web browsers' security model presented by Fett et al.[FKS14].

- CVSS (Common Vulnerability Scoring System) to score vulnerabilities on a defined set of metadata, and
- OVAL (Open Vulnerability and Assessment Language) which provides a declarative language to define tests for the presence of known vulnerabilities.

CVE and CVSS are pervasive, practically all vulnerability databases use them. While the goal of CVSS is to assign a score to vulnerabilities, we are interested in the metadata fields that are used to compute this score, as this is the only widely available meta data that describes the effect of vulnerabilities and the circumstances in which they are present in a machine-readable manner.

5.4.1 CVSS v3.1

The metadata defined in the latest version 3.1 of CVSS, released in June 2019, gives information about the difficulty of exploiting a vulnerability and the impact of such an exploit. The following fields define the prerequisites to an attack.

- *access vector*, which distinguishes vulnerabilities that are remotely exploitable from those that require local access
- *attack complexity*, which combines the probability of finding an exploitable configuration, the probability of a probabilistic exploit to succeed, and the skill required to mount the attack into either 'low', 'high'
- *privileges required*, which categorizes the necessary privileges to mount the attack into 'none', 'low' or 'high'
- *user interaction*, which is 'required', if the user needs to take action before the vulnerability can be exploited, and 'none' otherwise

The effect of a vulnerability is covered in terms of

- its *impact on confidentiality, integrity and availability*, either of which can be 'none', 'low', or 'high', and
- a change in *scope*, which a boolean that is set if, quoting

> "An exploited vulnerability can affect resources beyond the security scope managed by the security authority of the vulnerable component."

The scope metric was introduced with version 3 of the standard, released in 2015. Up to that point, the NVD did not provide data on how vulnerabilities may impact components other than the vulnerable component, e.g., in case of a privilege escalation. Privilege escalations were typically filed with integrity impact 'high'. The scope metric, however, did not solve the problem. First, it does not provide the necessary granularity. Second, the semantics of 'scope' is unclear to the point where even examples in the specification are disputed. The theoretical framework from the last section will allow us to argue the first point and provide a clear definition to remedy the second point.

5.4.1.1 Lack of Granularity

The most common reading of the standard suggests that hyper escalations have a change in ‘scope’, while local escalations and non-escalation have not. Local escalations are therefore encoded in terms of the integrity impact and are not distinguished from other vulnerabilities with integrity impact. For instance, a browser vulnerability that allows active content in one tab to manipulate content in another tab would have unchanged scope and high or low integrity impact. A browser vulnerability that allows altering the browser history is potentially marked the same.

5.4.1.2 Semantics

The semantics of the ‘scope’ field is unclear. First, there is no clear notion of what constitutes the ‘scope of the security authority‘ of a vulnerable component. An example in the CVSSv3.0 specification document considers a vulnerability in Microsoft Word that permits write access to files to have ‘unchanged scope’. Security experts pointed out that, indeed, Microsoft Word is a sandboxing mechanism [Ris17]. The categorization graph we propose defines a clear hierarchy that defines Word as an access control mechanism. Whether this vulnerability would be considered a local escalation or a hyper escalation would depend on which privileges can be gained and whether they are within the boundaries of Word’s security lattice. This is far easier to decide than what resources the vulnerability can have an effect on.

Second, the meaning of a resource is unclear and too wide. The specification explicitly considers XSS vulnerabilities to have a change in scope, because they affect a victim’s web browser by running code on it. But even a non-escalating vulnerability can affect a browser, e.g., by displaying an incorrect user name. But more importantly, by considering remote systems as components, it becomes impossible to instantiate the effect of a vulnerability on a given system. A change in scope might indicate an effect on the access control mechanism that controls the web application, but also *any other component in the network*. Instead, we propose to define the effect w.r.t. the local software stack, and relate local privileges by means of a separate relation, as outlined in Example 1. This relation can be locally computed, as we demonstrate in Section 5.7.

Third, and in the same vein, the meaning of the impact vector becomes unclear, because, according to the specification, it describes the effect on either the vulnerable component, or the affected component, whichever is worse. When estimating the damage on a given network, it is important to know *what* is affected, because both assets might have a vastly different importance.

5.4.2 A Refined Scope Field

We therefore propose a simple three valued field that has a much clearer semantics and greatly increases the ability of third-party tools to operate on this data with just a single bit of additional data. In the next section, we show that this field can be computed with high accuracy from existing data, providing a cheap mitigation path. Besides the benefits to the users, a clearer definition could lower the cost of tagging in the long run.

We propose this field to distinguish between the three categories from Section 5.3:

non-escalation, local escalation, and hyper escalation. This corresponds to the three operators $\perp, \uparrow$ and $\Uparrow$. Only the second comes with a loss of accuracy in our model. It is an over-approximation (see Corollary 1). Avoiding this loss of accuracy would require a system-independent description of the privileges gained, as well as those necessary. Such a description would have to be software-agnostic, but also allow for an instantiation of vulnerability's effect given a specific system. We are skeptical about the existence of such a language, and therefore advocate for a trading simplicity for accuracy at this point.

The impact vector of a non-escalation thus always concerns the vulnerable component, while the impact of a local escalation or hyper-escalation describes the nature of the additional privileges gained. For local escalation, these are controlled by vulnerable component. For a hyper escalation, they concern the privileges *of* the vulnerable component. Given a system, these possible effects can be instantiated (see Section 5.7).

5.5 Feasibility Study

We demonstrate the feasibility of annotating vulnerabilities with the type of escalation they allow by building a classifier that assigns these categories with high accuracy. This classifier uses the existing CVSS scores and other meta data available in the NVD. It can hence both serve as a plug-in solution to support risk estimation techniques that build on the NVD as it is, and as a proof of concept for a migration path to a new standard. Even with a small amount of manually annotated data, we obtain reasonable accuracy.

Our model performs multi-class single-label classification with each of the 3 classes representing one of the proposed annotations, $\perp$, $\uparrow$or $\Uparrow$. To obtain training data, we asked 5 CS students with experience in cyber security to annotate a total of 1151 vulnerabilities based on their NVD entry, which includes the above-mentioned meta data, but also textual descriptions and links to third-party websites. We randomly sampled 1151 vulnerabilities from the NVD and obtained the following distribution: 21% were tagged with $\perp$, 49% with $\uparrow$, and 30% with $\Uparrow$. The theory described in Section 5.3 was covered in a 30 min presentation followed by a discussion of a few example cases.

We decided to use a feed-forward neural network with 2 hidden layers after ruling out simple classification algorithms like support vector machines and k-nearest neighbors. The first hidden layer has 4096 neurons, the second 512 with each of them using relu as activation function. The feature vector was obtained through one-hot encoding a subset of the categorical variables in the CVE meta data, including the CVSS metrics. We also encoded the textual description using Google's Universal Sentence Encoder [CYK$^+$18]. We evaluated the performance of our model with a 10-fold cross validation, yielding an average accuracy of 87%. This accuracy is reasonable for a classifier that annotates existing NVD entries on the fly and permits immediate use of our privilege escalation model. Going one step further, we asked ourselves: what would be the cost of adopting our proposed change to the standard at the scale? Given a target accuracy, the goal would be to minimize the number of entries that need to be labeled by hand. We hence considered the softmax output of our model as a confidence measure to determine which entries need manual annotation. To this end, we set a threshold on the softmax output and computed both the *classification accuracy*, i.e., the accuracy on the set of vulnerabilities that are automatically classified, and the *total accuracy* assuming all manual annotations to be ground truth. The total accuracy is monotonically increasing by definition, but the classification accuracy might stay constant or even decrease (which would indicate that the softmax output is an unsuitable measure of confidence).

In Figure 5.5, we plot both measures as a function of the fraction of the dataset whose output score is below the threshold and hence needs to be manually annotated. Observe first, that the classification accuracy is, except for a small dent between 41% and 45%, monotonically increasing. Hence the confidence score, despite its known weakness in terms of interpretability [GPS+17], is sufficiently accurate to define the split. Second, the total accuracy reaches above 95% with 37% manually classified annotations. With 20 138 vulnerabilities filed in 2021, this would require the annotation of only 7 451 vulnerabilities. Based on a classification rate of 20 to 25 vulnerabilities per hour and personal cost for student staff (which is below industry average), a manual annotation cost us $ 1.4 per vulnerability. We expect classification accuracy to further improve with larger amounts of data and manual classification rates to improve over time, hence we consider these cost to be an upper bound. Under these circumstances, migrating the vulnerabilities filed to the NVD in 2021 with a target accuracy of 95% would cost $ 10 431. The top axis in Figure 5.5 gives an indication of the cost to reach other accuracy goals.

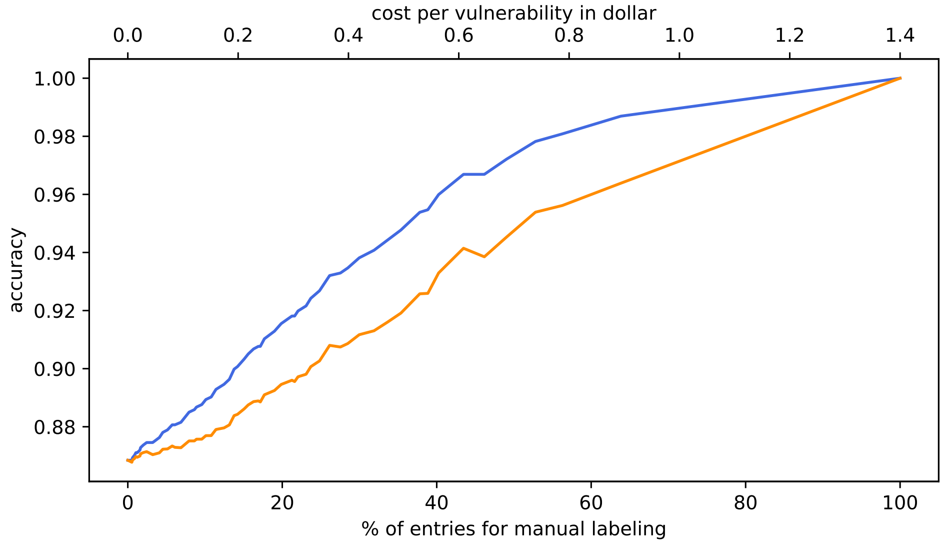


Figure 5.5: Classification accuracy (orange line below) and total accuracy per fraction of NVD that is manually labeled (blue line on top). The cost per vulnerability is linearly dependent on the latter and displayed in the top axis.

5.6 Planning Model

We use Automated Planning to find possible chains of (privilege escalation) vulnerabilities and other attacker actions.[5]

To aid administrators in selecting from the vast pool of possible countermeasures, and in a first step towards automated responses, we also want to compute and suggest the optimal countermeasures. We proposed *Stackelberg planning* in Part I to this end, formalizing a two-fold planning task — a *Stackelberg game* —, where the defender always moves first, while the attacker can fully observe the first player's action and gives the best response.

Stackelberg planning was already used for simulated pentesting and countermeasure selection based on Nessus reports [SSSH$^+$19]. The network attacker / follower player tries to control an important asset in the network. The defender / leader player modifies the network by applying countermeasures, trying to minimize the attacker's probability of reaching the goal while minimizing their own cost. We associate all defender actions with *cost* (which the defender minimizes) and attacker actions with *probabilities* (which the attacker tries to maximize). Given the network in either the initial state or some state resulting from applying a sequence of defender actions — a mitigation strategy — the *critical attack path* is the sequence of attacker actions whose success probability is maximal and reaches the goal. The *leader-follower search* from Section 3.3 computes the *Pareto frontier*, which contains all mitigation strategies that are Pareto-optimal. A mitigation strategy is Pareto-optimal, if there is no other that has strictly smaller defender cost and while keeping the attacker's success probability the same or lower, or vice versa.

The network model in [SSSH$^+$19] follows along the lines of MulVal [OGA05]. Just like MulVal, it processes Nessus reports and interprets vulnerabilities with integrity impact to give the adversary full control over the host. We enrich this model with the theory from Section 5.3 and use a custom acquisition toolchain to obtain security lattices and credential relations. Much like MulVal's Datalog-style rules, our planning rules can be represented in attack graph-style formalisms with probabilistic information.[6] We chose Stackelberg planning because it is ideal for fast prototyping and still performs reasonably well. It can handle up to 200 hosts easily [SSSH$^+$19], which, for countermeasure selection, is, to our knowledge, unmatched. Building on this model, we implemented the theory from Section 5.3 as additional predicates and action schemes.

5.6.1 Model

We sketch the most important parts of the planning model for the theory from Section 5.3 here. If you want to skip this, you can continue reading Section 5.7 on page 63.

Building on the model presented in [SSSH$^+$19], we present the most important predicates in Table 5.1 and the most important action schemes in Figure 5.6. Our notation presents the action's preconditions and postconditions as inference rules. We

[5] Our theory and acquisition techniques apply to all risk analysis techniques mentioned so far. For attack graphs, specifically, planning is equivalent, if only a single goal host needs to be reached [Hof15].

[6] Barring the probabilities, the attack rules are relaxed, i.e., no rule deletes a fact, and their postconditions are singleton. They are hence equivalent to Horn clauses.

Table 5.1: Planning predicates.

$\mathsf{credEx}(c, p)$	credential c accessible with privileges p.
$\mathsf{access}(p, p')$	privilege p allows to connect (over network) to software running with privileges p'.
$\mathsf{credMatch}(c, c')$	credential c matches c'.
$\mathsf{hasPriv}(p)$	attacker has privilege p.
$\mathsf{gainAccess}(p)$	implies $\mathsf{hasPriv}(p')$ for $p' \leq p$ (rules omitted).
$\mathsf{top}(p)$	privilege p is the top element ($\top$) in its security lattice.
$\mathsf{explExists}(\mathsf{cve}, p)$	exploit with id cve exists on software that runs with privileges p (in parent ACM).
$\mathsf{Hyper}(\mathsf{cve})$	cve classifies as a hyper escalation.
$\mathsf{Local}(\mathsf{cve})$	cve classifies as a local escalation.
$\mathsf{controls}(p, p')$	software with privileges p implements ACM whose security lattice includes p'.
$\mathsf{LoginMatch}(c, p)$	credential c contains login data for privileges p within some web service.

explain the action schemes one by one. The first type of attacker action is called A-RemoteConnect. It applies when the attacker has obtained some privilege p — recall that privileges are unique over all hosts and software configurations, hence they encode the host. If the privilege p affords the attacker access to some credential c which is used for authorization for some privilege p' (typically on a different host), then the attacker's access to p' follows as a postcondition. A typical use case is SSH credentials. MAUS sets $\mathsf{credMatch}$ (c,c') if c is an SSH key pair and c' a list of authorized public keys that contains the public part of c. The attacker actions A-ExploitHyper and A-ExploitLocal reflect the attacker's ability to exploit an existing vulnerability cve in some software identified by its privileges p'. If the attacker can connect to the vulnerable software with the current privileges p, he or she either gains privileges p' (if cve is a hyper escalation), or the top privileges p'' under the control of p' (if cve is a local escalation). The third type of attacker action is called A-UserAccess and is specific for browser-level privileges p_C. Here, the attacker controls server-side privileges p_S. From investigating a client's credential c, we can conclude that he or she regularly utilizes privileges p_S on the web server, which can thus be used to run code in the access control context of their browser. Note the similarity with A-RemoteConnect: by defining $\mathsf{credMatch}$ accordingly, we could alternatively define A-UserAccess as an instance of this rule. We chose not to do so, for clarity: in the current implementation, the server-side privileges p_S are not bound to p_C by matching credentials, but by the client-side login data. All the attacker actions listed here have a common postcondition,

$$\frac{\mathsf{hasPriv}(p) \quad \mathsf{credEx}(c,p) \quad \mathsf{credEx}(c',p') \quad \mathsf{credMatch}(c,c')}{\mathsf{gainAccess}(p')} \ (\text{A-RemoteConnect})$$

$$\frac{\mathsf{hasPriv}(p) \quad \mathsf{access}(p,p') \quad \mathsf{explExists}(\mathsf{cve},p') \quad \mathsf{Hyper}(\mathsf{cve})}{\mathsf{gainAccess}(p')} \ (\text{A-ExploitHyper})$$

$$\frac{\mathsf{hasPriv}(p) \quad \mathsf{access}(p,p') \quad \mathsf{explExists}(\mathsf{cve},p') \quad \mathsf{controls}(p',p'') \quad \mathsf{top}(p'') \quad \mathsf{Local}(\mathsf{cve})}{\mathsf{gainAccess}(p'')} \ (\text{A-ExploitLocal})$$

$$\frac{\mathsf{hasPriv}(p_S) \quad \mathsf{credEx}(c,p_C) \quad \mathsf{LoginMatch}(c,p_S)}{\mathsf{gainAccess}(p_C)} \ (\text{A-UserAccess})$$

$$\frac{\mathsf{access}(p,p')}{\neg\mathsf{access}(p,p')} \ (\text{D-FirewallRule})$$

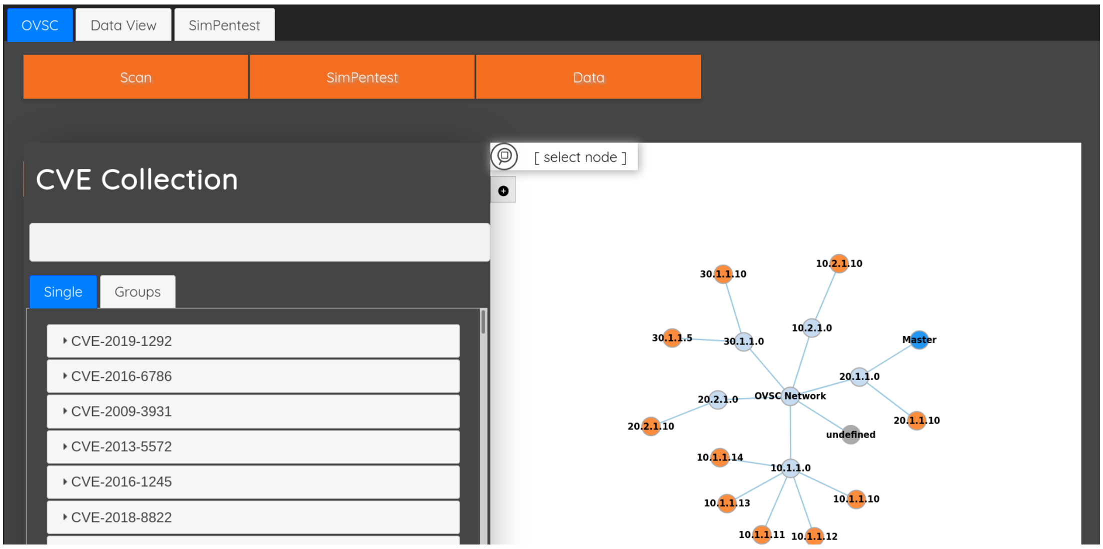
$$\frac{\mathsf{explExists}(\mathsf{cve},p)}{\neg\mathsf{explExists}(\mathsf{cve},p)} \ (\text{D-DelExploit})$$

Figure 5.6: Attacker and defender actions inference rules.

the predicate gainAccess. It signifies that the attacker gains access over the privilege p in the security lattice, implicitly obtaining all of its child nodes. We reflect this implication in our model with additional actions not listed here. The probabilities are set as follows: $p_{\text{A-RemoteConnect}} = 1$ and $p_{\text{A-UserAccess}} = 0.8$. The probabilities for A-ExploitLocal and A-ExploitHyper are derived from the CVSS exploitability metric of cve, which is not intended to denote a probability, but falls conveniently between 0 and 0.39 when divided by 10. We consider two types of defender actions. The first one is D-FirewallRule, which represents the ability of a system administrator to block a specific connection. This is reflected in our model by simply deleting the respective access predicate. The associated cost of this action can vary, because the modeler may want to reflect loss of functionality in terms of cost. The second is D-DelExploit. It represents the application of a software patch or the correct configuration of the affected software to mitigate a certain vulnerability. This is reflected by deleting the respecting explExists predicate. The associated costs can depend on the type of software because, e.g., legacy software requires custom patches. Finally, the initial state entails the instances of the predicates which define the unaltered network. The goal condition is that the attacker has the privilege over at least one of the important hosts which are provided by the user.

5.7 CATSS/MAUS

Our toolchain is divided into two components: CATSS (Centralized Analysis Tool for Security Scanning) and MAUS (Mitigation Analysis Using Simulation). CATSS builds on a manager-worker design to simplify network scanning from a central host. It improves on OpenVAS and Nessus's authenticated scanning mode in the following aspects. First, it keeps processing on worker hosts as non-invasive as possible by minimizing dependencies, relying only on SSH and libc/libstdc++. Second, CATSS uses a simpler and more extendible communication protocol based on JSON over TLS. Third, in contrast to NASL, the imperative language that Nessus and OpenVAS use, CATSS builds on a standardized declarative language for tests.



CVE Collection

- CVE-2019-1292
- CVE-2016-6786
- CVE-2009-3931
- CVE-2013-5572
- CVE-2016-1245
- CVE-2018-8822

Figure 5.7: CATSS/MAUS: After network discovery, the user can scan hosts and simulate a pentest.

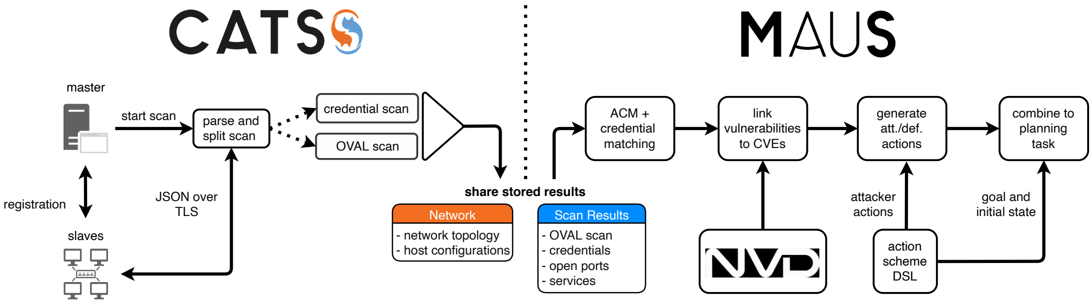


Figure 5.8: Architecture of CATSS/MAUS.

5.7.1 Open Vulnerability and Assessment Language

OVAL [BHH11] was developed by the MITRE organization. It can represent system information and specific machine states, e.g., a software inventory, installed patches and, of course, known vulnerabilities. It takes a declarative design approach and has standardized schemas for system information, machine states and test results. CATSS collects and parses OVAL definitions from local or remote sources. To speeding up network scans and to lower computational load, CATSS computes host-specific subsets, e.g., based on their applicability (determined, e.g., by OS version) and CVE metadata. The OVAL definitions are evaluated on the workers by transferring or building a binary that contains Red Hat's OpenSCAP [Ope], which locally evaluates the tests transmitted by the master host.

5.7.2 Credential Scanning

CATSS implements credential scanning by collecting credentials on all systems and comparing their hashes. Centralized key stores are still not widely employed; we therefore define an XML scheme that allows to manually specify credential tests using path schemes and/or content patterns. As a basic example, we implemented scans for SSH credentials by matching the files 'authorized_keys' and 'id_ed25519.pub'. The privileges necessary to obtain SSH credentials are obtained through the file permission on the latter.

5.7.3 Network Discovery

The scanning process is highly automated. In contrast to MulVal [OGA05] and the prototype [SSSH$^+$19], we are able to automatically discover which host is connected to which. To this end, CATSS performs pairwise network discovery using NMAP. Which services are running are determined locally, using lsof. Additionally we also scan installed packages to aid the process of matching services to vulnerabilities. The results aggregated by these major scan features serve as input to MAUS, determining our white box attacker's arsenal and the respective mitigation actions.

5.7.4 Mitigation Analysis Using Simulation (MAUS)

The second component, MAUS, generates a planning model from CATSS' scan results and network model. It instantiates the rule schemas depicted in Section 5.6 as follows. First, the OVAL scan data and vulnerability definitions are enriched with up-to-date meta information from the NVD. From the CVSS data, we gather the action probabilities, the impact on confidentiality and integrity (within the software's ACM), the impact on availability and whether the vulnerability can be remotely exploited, or requires local privileges.

The set of configurations S is represented as a tree of security latices, where the edge points to the security lattice containing the privileges of the current configuration. Each node hence corresponds to a piece of software identified during scanning and can be annotated with software-specific rules, allowing rapid changes to the model. At the moment, the software supports operating system level privileges; support for in-browser privileges and arbitrarily nested docker containers is in progress. Adding, e.g., support for gitlab, would entail a module that constructs the lattice for a given instance on a system, which would be transparently integrated when the algorithm traverses the corresponding node in the tree structure. Premier targets for adding support are database systems and web browsers.

5.8 Limitations

Simulated pentesting provides a principled way of providing risk analysis and mitigation analysis in networks. Like any formal method, it relies on an adequate model of reality and is bound to focus on certain aspects of it ('The map is not the territory'). The focus of the present model is publicly known, testable vulnerabilities. This excludes the following attack vectors. (a) Unknown vulnerabilities. Llanso and McNeil estimate the number of latent vulnerabilities in networks to exceed those that are known [LM], but note that zero-day *exploits* are rarely observed [BD12]. If the probability of a latent vulnerability can be estimated, a probabilistic rule can reflect this assumption. (b) Social engineering attacks. These can again be modeled using probabilistic rules, but existing models (e.g., [HMN15]) require knowledge about the users. Compliance to ISO 27001 can (depending on the level) entail an information security management system and defined procedures for security qualification, which may be sourced to obtain this data. (c) Shared cloud resources. The network is regarded as a self-contained system. Shared cloud resources can both allow for lateral movement and, albeit with much lower probability, become points of entry. We believe the first point can be covered by detecting common credentials or access histories, similar to how credentials are detected.

CATSS/MAUS have two major limitations. In contrast to vulnerability scanners like nmap or OpenVAS, CATSS requires full privileges to the system. This trades a vastly more precise model and the ability to control the load of production systems for a more complicated setup and the possibility of a system breach via the acquisition infrastructure. Furthermore, as more and more data is collected, the privacy of employees may become an issue. Our current prototype does not perform automatic network discovery, e.g., scanning for NAT, but it is able to test connectivity among known hosts.

5.9 Evaluation

In this section, we evaluate the use of our theory and the performance of CATSS/MAUS. We investigate the following evaluation questions:

EQ1 Does modeling privilege escalations lead to a noticeable improvement in the computation of the attacker's success probability?

EQ2 Is the acquisition of the data to build this model feasible for large networks?

EQ3 What is the effect of the larger model on the performance of the Stackelberg planning algorithm?

To answer questions EQ1 and EQ3, we compare our privilege escalation model with a simplified model without local privilege escalations and shared credentials, which is very similar to those of the previous tool [SSSH^{+}19] and MulVal [OGA05].

5.9.1 Deployment Infrastructure

A major difficulty in our evaluation is the lack of a widely accepted benchmark suite. Ideally, there would be a small set of networks that is representative of the topology and vulnerabilities of networks in the wild. In contrast to program code, it is not easy to put networks in a package. Furthermore, it is difficult to obtain the permission to publish data about real-world networks. Finally, even if there was a benchmark derived from real-world networks, it is unclear how representative those are (whereas, e.g., open-source tools like GNU binutils are representative already by how widely they are deployed, in addition to the considerable number of contributors). As a consequence, many experiments in simulated penetration testing lack reproducibility and their generalizability is unclear.

We highlight that our evaluation suffers from the same lack of generalizability. We devised, however, a methodology that allows to *reproduce* our findings and rapidly devise new example networks. At the very least, this will allow other researchers to evaluate the validity of our findings. It also facilitates extending the benchmark with new networks and thus paves the way to finding a benchmark that is agreed to generalize.

To package networks, we combine GNS3, an open-source network simulator, with nested docker containers. GNS3's origins are in (network) hardware virtualization, but recent versions support access to docker containers. First experiments with virtual machines and VIRL images of CISCO routers revealed that hardware virtualization scales only to a few dozen hosts and that managing their configurations was a major impediment. We improve on both points. To scale to large networks and to make the experiment easy to deploy, we nest two docker instances. In the outer layer, docker-compose is combining four containers, one simulating the network simulation, one running GNS3's GUI, one monitoring performance indicators and one running the CVE classifier from Section 5.4.2 as a service. The container running the network simulation contains yet another docker instance which runs containers for hosts that appear in the network. In addition to these containers, there is a process running GNS3. GNS3 simulates the network's topology and the communication between the hosts. Packets are relayed to the (nested) docker containers, which simulate each host's internal processing. In addition to these hosts, CATS/MAUS is running as an appliance within the simulated network. This architecture makes it possible to reproduce the experiment with a single invocation of docker-compose up and to deploy it in the cloud when a single system runs out of memory.[7] For the two case studies we will present in the follow-up, the simulation required less than 28 GB and 70 GB of memory, respectively. To facilitate the creation of networks we have devised a custom configuration format that allows the rapid creation of host configurations by combining a base image (e.g., a standard Linux install) with a given set of software configurations (e.g., a Firefox installation, or web server with some configuration) to a docker image.

5.9.2 Evaluation Metrics

We measure the performance of CATSS/MAUS in terms of the time it takes to scan a network, to translate the collected data into a planning problem and for the time used to solve the planning task. While the first two steps are parallelized, the solver is single-threaded. The runtime of the solver splits into a pre-computation phase, where some actions are filtered by a static relevance analysis and the actual full search for the Pareto frontier. Note that the translation already performs a partial grounding of actions, i.e., instantiates the action schemes with all applicable objects.

We measure the attacker search for the initial state, i.e., the computation of the critical path as well. This computation is performed at the start of every full search.

We, furthermore, measure the memory consumption, but only for the solving step, as the scan and translation phase are not memory intensive at all. We finally report

[7] See https://github.com/networklab2020/lab.

Table 5.2: Execution times for four phases, maximal memory use in solver, model size and probability of critical path $P_{critical}$.

				solve (FastDownward)				model size		
case study	model	scan	transl.	pre-comp.	full search	peak memory	attacker search	attacker	defender	$P_{critical}$
1	priv.	313s	50.5s	4.86s	0.01s	4MB	0.01s	53	190	39.0%
1	simple	309s	49.8s	4.68s	0.00s	4MB	0.00s	27	190	5.9%
1b	priv.	—	—	—	1 211.6s	4 489MB	1.71s	53	189	39.0%
2 (n=25)	priv.	957s	65.4s	9.83s	0.49s	11MB	0.27s	145	1402	15.2%
2 (n=25)	simple	937s	61.3s	9.87s	0.02s	6MB	0.00s	64	1402	5.9%
2 (n=40)	priv.	1 669s	79.7s	11.91s	0.09s	12MB	0.02s	208	3504	12.2%
2 (n=40)	simple	1 616s	74.2s	11.25s	0.05s	9MB	0.00s	91	3504	5.9%
2 (n=50)	priv.	2 307s	98.0s	16.68s	5.7s	35MB	1.77s	260	5828	31.2%
2 (n=50)	simple	2 288s	93.8s	15.34s	0.08s	11MB	0.00s	113	5828	5.9%

the model size in terms of the number of actions MAUS generated. This gives a rough indication of the size of the problem, but note that the performance is heavily influenced by the structure of the problem.

It is impossible to establish a ground truth for the probability of an attack on the network, hence we report the improvement in model accuracy w.r.t. prior work. We then compute the probability of the critical path, i.e., the path from X to Y in a network without new defenses in place, i.e., the status quo. The critical path tells us the action sequence with highest probability leading the attacker to the goal. We compare this number for the full model including privilege escalations, and to the same planning model stripped from all rules about privilege escalation and credential sharing. This model corresponds to the models used in earlier work [SSSH$^+$19; OGA05]. Our results are displayed in Table 5.2.

5.9.3 Case Study 1

The first example (see Figure 5.9) considers a network of 10 hosts within three subnets: a sensitive area where critical data is hosted, a user area where (potentially vulnerable) desktops reside and a DMZ that hosts a web server, an email server and a database server. Multiple vulnerabilities in the DMZ allow for compromising the hosts remotely. They open up possibilities for lateral movement towards the user area or the sensitive area. Some hosts in the user area store SSH credentials that provide access to the sensitive area of the network. The structure of the network allows the defender to completely eliminate lateral movement by blocking connections between the subnets, but these defender actions are, of course, associated with a higher cost due to the imposed restrictions on the capabilities of the hosts.

Vulnerabilities in the DMZ and credentials stored on user hosts provide access to the sensitive area of the network. The attack can be thwarted by blocking connections between the subnets or by updating software. Both types of measures are represented as defender actions (see Section 5.6). Blocking connections comes at a higher cost, as they restrict the capabilities of the hosts.

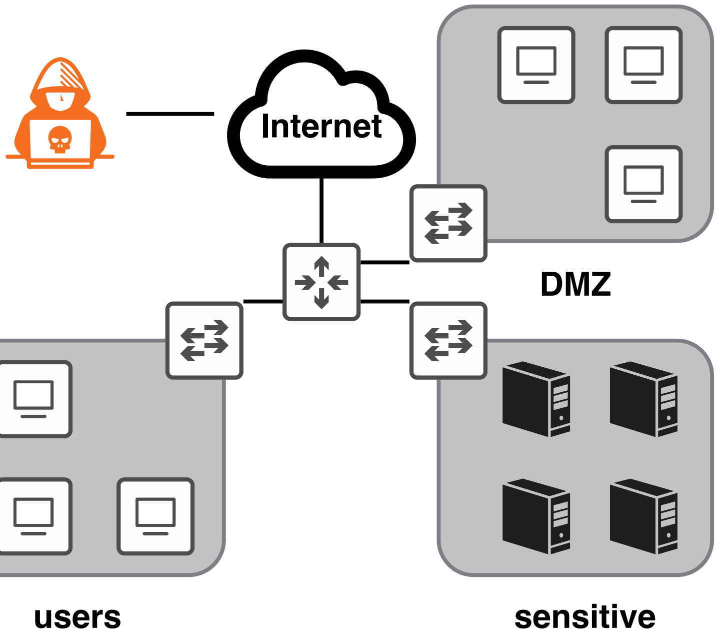


Figure 5.9: Architecture for case study 1.

This case study was hand-crafted to resemble a real network, following CISCO's design guidelines [Cis]. The configurations were based on an outdated Linux distribution[8], so they contain more vulnerable software. As mentioned before, these results cannot be assumed to generalize, and, in contrast to case study 2, there is no defined procedure to generate random instances of this network, so the reader cannot be sure it was not handpicked.[9]

5.9.3.1 Performance and Model Size

There is almost no performance overhead in scanning and translation . For scanning, this is because credential scanning runs in parallel to the OVAL scans, which tend to take longer. For translation, in general, the runtime depends on the host configuration in question, but there is little difference for this particular network. There is a significant overhead in the search time of the solver, but for a network of this size, precomputation (e.g., grounding actions) dominates the runtime. The privilege model has, expectedly, more attacker actions, but the same number of defender actions. To have a fair comparison, we optimized the simplified model. Given that we are ignoring the context of the security lattice, we can group CVE exploits solely by their attack vector and consider only the vulnerabilities that maximizes the attack probability. This greatly reduces the amount of attacker actions that are produced in the final domain file.

[8] Debian 8 (Jessie), released 25—26 April 2015.

[9] We do remark that the setup of a testing network takes considerable effort, so manipulation is costly, at least.

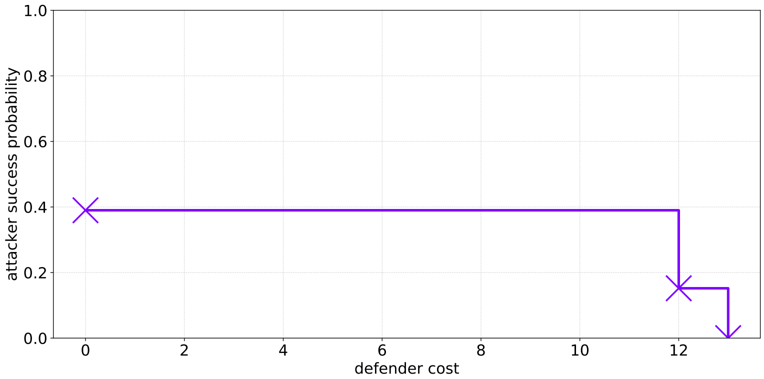


Figure 5.10: Pareto frontier for case study 1b.

5.9.3.2 Critical Path

In the privilege model, the probability of the critical path is much higher, because the attacker can progress very early by means of a credential relationship via SSH keys. These actions have an attack probability of 1.0, as we expect the adversary to exploit them whenever possible. In the simplified model, success probability decreases with the length of the attack sequence, as it has to rely on exploits instead. For this network, a chain of 3 exploits has to be combined to reach a target host in the sensitive area.

5.9.3.3 Case Study 1b

When investigating the Pareto frontier in case study 1, we noticed that the only non-empty defender-optimal strategy is to patch the vulnerabilities on two hosts in the DMZ and thus cut-off the adversary's ability to enter the network. In these situations — which are frequent in practice — the pruning techniques in the Stackelberg planning algorithm lead to a considerable reduction of the search tree. To challenge the algorithm and to find out whether there are more complex Pareto frontiers, we removed the ability to fix the vulnerability for one of these hosts in case study 1b. The resulting Pareto frontier (Figure 5.10) has two optimal defender strategies. The first includes 12 firewall rules; the second includes the same firewall actions and one additional D-DelExploit action. This leads to a remarkable difference in search time, with more than 20 minutes, compared to 10 ms for case study 1.[10]

5.9.4 Case Study 2

The focus of the second case study (see Figure 5.11) is on the performance of CATSS/-MAUS and Fast Downward with increasing network size, i.e., EQ2 and EQ3.

[10] While this may seem surprising at first, the leader-follower search is PSPACE-complete, scales exponential in the number of countermeasures, and is only tamed by exploiting various optimizations.

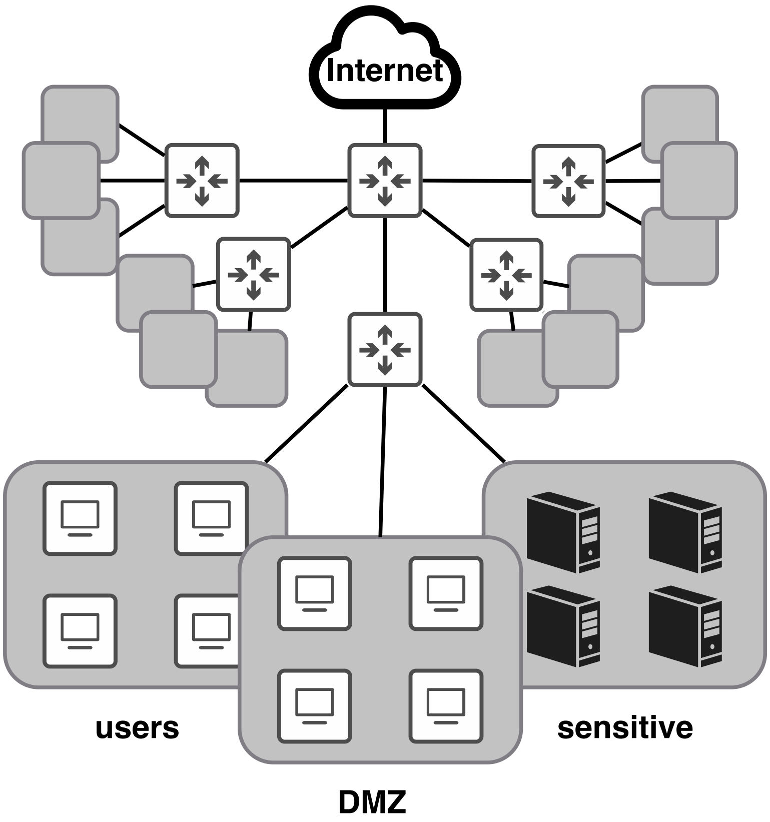


Figure 5.11: Architecture for case study 2 (omitted subnets from other departments).

5.9.4.1 Problem Generation

To uniformly scale the network size n, we proceeded as follows. For any n, the network is divided into 5 departments, each of which has a DMZ, a sensitive area and a user area, like in case study 1. Given a target size n, we

1. chose one of the five departments (uniformly) at random.

2. To keep the user area (U) at roughly the same scale as the sensitive area (S) and the DMZ (D) combined, we use a categorical distribution where $p_S = p_D = 0.25$ and $p_U = 0.5$.

3. We randomly chose a configuration within the given category. There are 4 configurations for U, 2 for S and 3 for D.

4. Credentials are not manually assigned, but are part of the configuration. The configurations in U give access to configurations in $S \cup U$.

5. We insert this configuration in the subnet determined by the first and second step.

5.9.4.2 Performance and Model Size

As expected, there is little difference between both models w.r.t. scanning performance. In absolute terms, the scanning time grows quadratically with the size of the network. This is an artifact of our modeling environment: computing the connectivity requires each host to communicate with each other. In a real network, this step runs in parallel on each host and we would expect it grows linearly with the size of the network. GNS3's simulation needs to linearize the communication, hence the quadratic growth.

In a real network, we expect the translation, in particular the parsing and aggregation of OVAL tests on the manager to dominate the runtime of CATSS. We see that the runtime of the translation scales linearly and is acceptable even for large networks.

The search time of Fast Downward grows with the network size, but is still low enough to be dominated by the precomputation time. On most problems, the ration between the search time of the privilege model and the simple model is between one or two orders of magnitude, similar to case study 1. The problem for $n = 40$ seems to be an exception, where the runtimes are roughly similar. The memory consumption grows with the size of the model, but is still less than 36 MB, even for $n = 50$. Here, the ratio between the privilege model and the simplified model is much lower and ranges between 2 and 4.

5.9.4.3 Critical Path

Again we observe a considerable difference in the probability of the critical path for all n. For the privilege model, the concrete value ranges between 12.2% and 31.2%. A qualitative analysis of the critical path shows that all types of attacker actions are actually used to reach a goal, and that the attack path is different for different networks. In the simple model, on the other hand, all critical paths have the same probability. Closer investigation shows that even the smaller cases have at least one department where all three areas contain a vulnerability with high exploitation probability. This is due to the low number of configurations. The default firewall rules already disable the communication between the DMZ and the sensitive area. Furthermore, the Internet can only access the DMZ. Without shared credentials, the shortest path from the Internet to a goal host is therefore via the DMZ and the user area to the sensitive area. The critical path results therefore from the exploitation of three vulnerabilities with maximal probability in the same department which happen to exist in every problem.

5.9.5 Summary

We can now come back to the aforementioned evaluation questions.

EQ1: The attacker's success probability is noticeably higher in the privilege model, at least in our case study. Within the boundaries of our case study, the attacker probability is noticeably higher in the privilege model. As the additional rules model adversarial capabilities that actually exist, we interpret this as a more accurate estimate. We stress, however, the limited statistical significance of our case study due to the low number of unique configurations.

EQ2: Even for large networks, it is feasible to acquire the data necessary to build the privilege model. Case study 2 shows that the translation takes less than 2 minutes, even for the largest network. The time for scanning grows quadratically due to our simulation approach, but should remain comfortably linear in a real network, as the scans run in parallel.

EQ3: The Stackelberg planning algorithm is sufficiently fast to handle the privilege model. The runtime of the mitigation search algorithm grows with the network size, but much less than expected given the worst-case runtime, which is exponential in the number of defender actions. It remains under six seconds even for the largest network (except for case study 1b which we crafted specifically to challenge the algorithm).

Limitations As discussed earlier, these experiments may not generalize, and it is not clear how to achieve generalizability at all. Instead, an open competition could help to define a set of problems that creates a common ground for comparison. We built even the larger networks from a set of nine configurations, which limits the validity of our findings concerning EQ1. Finally, the downside of our simulation approach is that the available cloud resources impose a limit on the size of the experiment. Our simulation ran on a VM with 32 CPUs and 128 GB RAM. Experiments with 100 hosts ran out of memory.

5.10 Conclusion

We proposed the first comprehensive theory for privilege escalation and demonstrated its use in improving formal threat models. It exposes a hierarchy of hierarchies that can be used to categorize vulnerabilities and instantiate the effect of exploits precisely in many cases. We demonstrated that it is both cheap and useful to augment the well-established CVSS standard by an additional field and obtain a clearly defined semantics for this field. These meta data have to be produced for more than ten thousand vulnerabilities per year and are used by a plethora of security products.

Our own security scanner and risk analysis tool, CATSS/MAUS, provides the first publicly available[11]tool to perform on-system scans and thus produce detailed network models. Its modular structure provides a basis for new risk estimation techniques.

Capitalizing on our Stackelberg planning foundations, we implemented our theory in MAUS and evaluated it for model quality and performance. The results indicate a low performance impact and improvement in the completeness of the model. A by-product of this evaluation is a method that allows to reproduce these experiments.

[11] We plan to make the tool publicly available for non-commercial use.

Part III

Applying Stackelberg Planning to Internet-Scale Applications

Applying Stackelberg planning based algorithms for solving Internet-scale mitigation analysis tasks

6 Formally Reasoning About the Cost and Efficacy of Securing the Email Infrastructure

6.1 Motivation and Contributions

The Internet infrastructure relies on the correct functioning of the basic underlying protocols for routing and name resolution, which, historically, were designed for functionality, not for security. Over the past decades, the security research community has put a lot of effort into strengthening the security of these protocols by applying cryptographic means on various layers of communication (e.g., IPsec, DNSSEC, TLS). But only a small fraction of them have been deployed and have a real-world impact. Is inertia the only reason that this is the case? For email alone, there are a multitude of proposals to mitigate known vulnerabilities, ranging from the introduction of completely new protocols to modifications of the communication paths used by big providers [AAL+05a; R A05; AAL+05b; Hof02; DH15; MRL+16; Res00]. At the very core, every meaningful decision about which measure to deploy calls for a deep and rigorous understanding of the induced benefits, the cost and the resulting effects.

In this work, we provide a thorough and automated methodology for making formal deployment assessments for various layers of the email infrastructure. We propose such a methodology in the form of a *mitigation analysis*. Our methodology models and analyzes the impact of different mitigation strategies against an attacker as a Stackelberg planning task. We adapt the model and analysis presented in Part I to Internet infrastructure attack mitigation, over a data set spanning more than 6 million domains, IPs, autonomous systems and DNS zones. Given the scale of the data set, the computational challenge is tremendous, as the best attacker strategy needs to be considered *for every combination of mitigation choices on the part of the defender.* Tackling this complexity is where security meets AI: following Part I, we employ Stackelberg planning methods for effective search in mitigation-choice space, geared at finding good choices quickly and pruning later choices against the bounds thus identified. Finally, the outcome of the mitigation analysis is the *Pareto frontier* of mitigation strategies, i.e., the optimal choice of mitigation efforts per budget. In practice, this provides a function from a given budget to the set of most effective mitigation strategies. Such an assessment is useful 1) for standardization bodies to estimate the potential improvement of the adoption of a new security standard, 2) for government bodies to evaluate strategies for digital self-reliance, 3) for protocol designers to guide and evaluate their decisions w.r.t. the current infrastructure, and 4) for providers to make investment decisions on their infrastructure.

We demonstrate our methodology at scale in a case study where we take the high-level perspective of a government body seeking to secure its Internet infrastructure. In particular, we focus on the case of email, which is used by 3.7 billion users world-wide [The17]. Since the Snowden revelations have unfolded, governments and the citizens they represent are aware of large-scale spying programs targeting ‘nearly everything a typical user does on the Internet’ [Gre13].

This is achieved by deep packet inspection (XKEYSCORE [Gre13]) and the cooperation of large American providers (PRISM [Lee13]).

Many governments realize the threat this poses for national security [BHR$^+$12], but also for trade secrets [fNSA15] and human rights [Har15], and have lately responded by creating various forms of organizations, most often military entities, that deal exclusively with security from cyber-attacks [OTG$^+$16; OCo17]. Given publicly available information and ballpark cost estimates for mitigations, we assess the cost and impact of mitigations in nine cost scenarios and 45 combinations of defender and attacker countries. Our results suggest a nuanced view. There is no single cure for all, viable mitigation strategies differ from country to country and scenario to scenario. Some might consider the idea of a country feeling responsible for the privacy of its citizen’s emails an overly idealistic view. However, given that countries under the jurisdiction of the European court of human rights are bound to the Convention for the Protection of Human Rights and Fundamental Freedoms, articles 6, 8 and 10 of which were found to be ‘endangered’ by ‘mass surveillance and large-scale intrusion practices’ [Har15], this goal is arguably a core responsibility of a constitutional government: the enforcement of basic constitutional human rights.

To summarize, we make the following contributions:

- We present a comprehensive deployment cost estimation methodology, reasoning about the cost versus the efficacy of different mitigation strategies (Section 6.2).
- We present a novel, formal threat model covering attacks on the routing level, name resolution and email communication (Section 6.3).
- We formalize the effects and estimate the cost of proposals like IPsec, DNSSEC, DANE, SMTP STS and SMTP over TLS (SMTPS), yielding a defender model composed of mitigation strategies for minimizing the attacker’s objective (Section 6.4).

- We apply our methodology on the German email infrastructure in the face of large-scale email sniffing and discuss different cost scenarios (Section 6.7.2), and 45 combinations of attacker and defender countries (Section 6.7.3).
- We identify viable mitigation strategies in different scenarios, as well as strategies that are only rarely worth the effort, while also highlighting deployment issues of existing approaches and their limits in the face of foreign infrastructure.

Limitations Our analysis relies on (i) a formal threat model and (ii) a formal defender model. Our threat model assumes the protocols covered here to work as intended; even if this is the case, as of now there is no formal justification for it to be sound and complete, e.g., w.r.t. a Dolev-Yao attacker and a correct implementation of the respective protocols. As we compare attacks by their effectiveness, both soundness and completeness are required. Our defender model depends on a realistic cost assessment. Given the scope of this chapter, we can provide only a rudimentary analysis that sacrifices precision for uniformity and clarity. We made an effort to provide sources for our cost assessment where possible, but point out that (a) cost vary from company to company and often depend on company secrets (b) our cost model includes only direct monetary cost (c) we do not quantify the margin of error of our results.

6.2 Planning

Our mitigation analysis is based on *Automated Planning*, more specific, Stackelberg planning.

Planning has already been successfully applied to network security and penetration testing tasks[BGH$^+$05; LSR10][SBH12; DL14], [Hof15]. This branch of research – network attack planning as a tool for automated security testing – has been called *simulated pentesting*. We introduced Stackelberg planning in Part I and applied it to local computer networks in Part II. We adopt the same approach here to our email infrastructure context.

6.2.1 Mitigation Analysis

Mitigation analysis through Stackelberg planning in the context of network penetration testing can be seen as a two-fold planning task, in which a defender plans for mitigation strategies that impose a limit on the worst any attacker can do.

Formally, the state of the world and the state of an attack are described through a finite set of propositions. A state is a truth value assignment to these propositions. Concrete attacks depend on the initial world state and are determined from a finite set of *attacker actions*. Associated with an attack is the *attacker reward* $\mathcal{R}$, which is used as an indicator of the severity of the attack. For example, as shown in Section 6.3.1, in our model the reward of an attack corresponds to the number of connections that could be compromised via the attack. To prevent attacks and thus to lower the attacker reward, the defender can change the world state through the application of *defender actions*.

In logical terms, an action is given by a *precondition pre*, a boolean formula over proposition literals and a *postcondition post*, a conjunction over proposition literals, and

```
 1) Reward = 0.217100, Cost = $0
    ∅

 2) Reward = 0.161200, Cost = $366 342
    ¬nDNSSEC(mail.ru)

 ⋮
15) Reward = 0.010400, Cost = $168 372 779
    ¬nDNSSEC(rambler.ru), ¬nDNSSEC(mail.ru), ¬nDNSSEC(yandex.ru),
    ¬nReloc(gmail.com), ¬nRFC7817(yandex.ru), ¬nRFC7817(mail.ru)

16) Reward = 0.000000, Cost = $186 404 888
    ¬nDNSSEC(rambler.ru), ¬nDNSSEC(mail.ru), ¬nDNSSEC(yandex.ru),
    ¬nReloc(gmail.com), ¬nRFC7817(yandex.ru), ¬nRFC7817(rambler.ru),
    ¬nRFC7817(mail.ru)
```

Figure 6.1: Excerpt of the Pareto frontier from USA vs. Russia.

is written as

$$\frac{pre}{post}.$$

An action may only be applied in states that satisfy the action's precondition. In the state resulting from this application, all propositions with positive occurrence in *post* are made true, and vice versa, all propositions with negative occurrence in *post* are made false. Additionally, every defender action is associated with a positive real *cost* value.

In the concrete attacker-planning model that we derive from our dataset (detailed in Section 6.3), we employ *monotonic* attacker actions, with positive preconditions and postconditions only (no negated literals). This corresponds to a widespread assumption in automated attack analysis, specifically attack analysis via *attack graphs* (e.g. [PS98a][TL00; AWK02], [GG09]): during the course of a given attack, the attacker incrementally gains new assets, but never loses any assets.

We compare different *mitigation strategies*, i.e., sequences σ of defender actions, in terms of their cost $c(\sigma)$: the sum of the cost of the defender actions in the sequence; and the impact on attacks $\mathcal{R}(\sigma)$: the attacker reward in the state resulting from the application of σ. We say that a mitigation strategy *dominates*, 'is better than', another mitigation strategy if its cost is strictly smaller while the attacker reward is not larger, or if the maximal attacker reward is strictly smaller while the cost is not larger, i.e., either $\mathcal{R}(\sigma) \leq \mathcal{R}(\sigma')$ and $c(\sigma) < c(\sigma')$, or $\mathcal{R}(\sigma) < \mathcal{R}(\sigma')$ and $c(\sigma) \leq c(\sigma')$. Our analysis yields the *Pareto frontier* $\mathcal{P}$ of mitigation strategies: the set of mitigation strategies which are not dominated by any other mitigation strategy. Figure 6.1 gives an example of a Pareto frontier output by our analysis tool, the details of which will become clear in Section 6.7.

6.2.2 Planning Algorithm

To compute the Pareto frontier, we extend the *leader-follower search* algorithm introduced in Section 3.3 to our setting.

In Part I, we already proposed several pruning techniques to skip mitigation options which are irrelevant or inefficient considering the current network state and the mitigations considered so far. We adopt these techniques here as well, but even with these optimizations, analyzing data at Internet scale would be far beyond the scope of the algorithm. To make it feasible to run this mitigation analysis approach on our model, we identify parts that are relevant for neither the attacker nor for the defender, and that can thus be removed prior to the analysis. The identification of irrelevant parts here is based on the concept of *property graphs*.

6.2.3 Property Graph

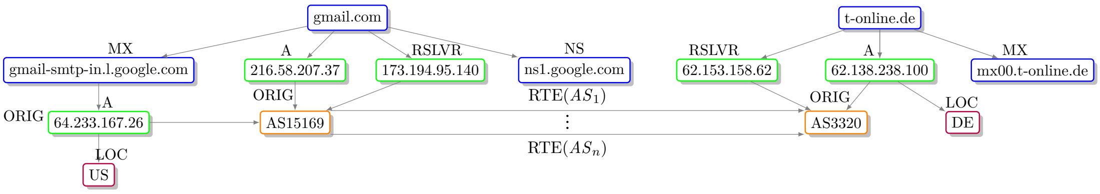


Figure 6.2: Snippet of the property graph.

A basic input to our planning model machinery is the property graph, a labeled graph introduced by Simeonovski et al. [SPR$^+$17b]. Nodes in this graph represent IP addresses, domain names, Autonomous Systems (AS, a group of routers whose addresses and routing policies are under common administrative control) and countries; edges represent relationships between them.

We extend the graph formalism from prior work with AS-level routes and an indication of DNS resolvers used. The AS-level routes indicate the ASes that a packet traverses when transmitted from source to destination. These are directed, as routes are not always symmetrical. The resolver, or local DNS resolver, is the last recursive name server that performs iterative domain resolution. We will explain how the DNS resolver is identified and the routing data collected in Section 6.6.3.

Figure 6.2 presents a snippet of the property graph. Nodes in the set Dom indicate domain names, e.g., gmail.com with a DNS record A mapping the IP addresses (nodes in the set IP), e.g., 216.58.207.37, which in turn belongs to an AS (AS15169) and is geolocated in the US. A directed first-order edge RTE(AS_t) (AS_t is a property of this edge) indicates that there is *some* route from the source AS to the destination AS which traverses AS_t.

For instance, in our example the edge labels show that AS_1 to AS_n might be traversed if a user using gmail.com sends an email to t-online.de. The set of labels $\{AS_1, \ldots, AS_n\}$ is the union of ASes that appear in some route, but not a route itself, as we only need data at this granularity. The graph does not include peering agreements between ASes. Given these agreements and each AS'es routing policy, these routing edges can be computed; we choose to probe routes used instead. Table 6.1 summarizes

Table 6.1: Labels of nodes and relationships.

node/edge	description
IP	IP address.
Dom	domain name.
AS	IANA number assigned to the AS.
Cntry	country code.
ORIG	*AS* where an lhs originates from.
LOC	*country* where lhs (IP ∪ Dom ∪ AS) is located.
DNS	resolving lhs domain requires query to rhs.
RSLVR	lhs uses resolver on rhs for name resolution.
A	DNS record mapping Dom to IP.
MX	DNS record mapping Dom for email delivery.
RTE(AS_t)	AS-level route via AS_t between two $ASes$.

the types of nodes and edges which we consider in our model. The threat model described in the next section is formulated in terms of rules with predicates of form $a \xrightarrow{B} c$ or $a \in S$ for the existence of labeled edges and node types in the property graph.

6.2.4 Precomputation

Our analysis determines the relevant part of the graph by combining a tainting similar to the approach proposed by Simeonovski et al. [SPR+17b] with a reversed tainting starting from the email providers. Only nodes tainted from both directions need to be considered, significantly reducing the size of the graph relevant for the analysis. More precisely: given the full graph, all attacker rules are applied until a fixed point is reached. These are monotonous, i.e., the order in which they are applied is unimportant (cf. Section 6.3). All nodes and edges appearing in any instantiation of a rule that was applied are 'forward tainted'. For any provider that was forward tainted, we proceed in the opposite direction, using only actions that were applied in the forward tainting. Interpreting them backward, we again apply them until a fixed point is reached. Nodes and edges appearing in these actions are now forward and backward tainted and define the relevant sub graph of the problem.

6.3 Threat Model

Since the Snowden revelations have unfolded, governments are aware of and react to large-scale spying programs targeting 'nearly everything a typical user does on the Internet' via deep packet inspection and cooperation of domestic companies [Gre13]. Hence our threat model considers an infrastructure attacker who is able to use routing and DNS spoofing to mount attacks which are not specific to implementation vulnerabilities or protocol weaknesses of SMTP, but concern the underlying assumptions on routing and domain resolutions that the security of protocols without endpoint verification, such as SMTP, rely on. Furthermore, the attacking country has the means to compromise servers located in its jurisdiction, e.g., using gag orders or legislature, and to observe and intercept packets routed via ASes under its jurisdiction. The goal of the attacker is large-scale surveillance, hence the attacker is active, but restricted to easy-too-mount attacks which scale well and avoid global exposure. For this reason, we do not consider off-path attacks on DNS [Kam08] or false BGP announcements.

In contrast to formal protocol verification, our threat model considers the combination of known attack vectors w.r.t. routing, name resolution, and email communication in a more abstract, attack-centric view. The attacker is not in full control over the network but can inject packets at certain routes.

We ignore client-specific defense measures such as end-to-end encryption because they are rarely used. PGP, arguably the most prevalent of those, has never gained enough traction to prevent large-scale attacks: currently 4.7M keys are within the SKS key pool [Web17], and much less are part of a strongly connected set [Pen17].

6.3.1 Attacker Reward

The goal of the attacker is to observe as much of the defender country's email communication as possible. Instead of counting the number of email domains the adversary can gain control of, we estimate the impact in terms of users affected based on market share data and the total number of Internet users in the defending country. Not only does this provide us with a meaningful impact metric, but it facilitates the analysis, as in most countries email is a market with very few players. We represent those as a set of domains $\mathsf{Provider} \subseteq \mathrm{Dom}$ and consider only providers for which market share data is available. The ability to intercept emails from provider A to provider B is represented by a predicate $\mathsf{unconf}(A, B)$. The attacker optimizes the number distinct tuples A, B, weighted by their respective share of communication $\omega(A, B)$, which is the product of the market share of A and B: $\mathcal{R} = \max \sum_{d,e \in \mathsf{Provider}\text{ s.t. }\mathsf{unconf}(d,e)} \omega(d, e)$[1].

6.3.2 Attacker Actions

We formalize the threat model as a set of attack rules, which, when instantiated with all domains, countries, ASes, IPs, etc. give us a large but finite set of attacker actions. The complete set of rules, including intuitive descriptions, is given in Section 6.5.1. Here we will focus on the most interesting cases. All attacks we present here are well known, the contribution of this section is limited to their formalization.

We distinguish between domains, IPs or ASes that initially belong to the adversary, domains that do not belong to the adversary, but can be resolved to adversarial IPs, and MXes which can be resolved to adversarial domains. This corresponds to direct (initial) compromise in the real world, compromise on the DNS layer, and compromise on the application layer, i.e., email (which relies on DNS). Similarly, we distinguish between compromise of communication on the routing, DNS, and application layer. Table 6.2 gives an overview of the predicates we use in our modeling.

[1] A concrete interpretation can be given as follows: if every email user in the defending country sends an email to every other user (technically including themselves, but the error is negligible), the reward equals the percentage of emails the attacker can read.

Table 6.2: Integrity and confidentiality impact predicates.

$\mathsf{C}(x)$	Node $x \in \text{Dom}\cup\text{IP}\cup\text{AS}\cup\text{Cntry}$ under adversarial control.
$\mathsf{I}^{\mathsf{DNS}}(d)$	Integrity of name resolution of $d \in$ Dom compromised
$\mathsf{I}^{\mathsf{R}}(i,j)$	Integrity of *some* route from $i \in$ IP to $j \in$ IP is compromised.
$\mathsf{I}^{\mathsf{DNS}}(d,e)$	Integrity of name resolution of $e \in$ Dom from perspective of $d \in$ Dom compromised.
$\mathsf{unconf}(d,e)$	email communication going from some user of $d \in \mathsf{Provider}$ to some user of $e \in \mathsf{Provider}$ is considered unconfidential.
$\mathsf{nDNSSEC}(d)$	$d \in$ Dom does not support DNSSEC.
$\mathsf{nTLS}^{\mathsf{snd}}(d)$	$d \in$ Dom does not enforce strict host validation when sending emails.
$\mathsf{nVPN}(a,b)$	communication between $a, b \in$ AS is not secured with IPsec.
$\mathsf{nDANE}^{\mathsf{rcv}}(d)$	$d \in$ Dom does not support DANE.
$\mathsf{nRFC7817}(d)$	$d \in$ Dom does not validate the receiving server's domain part of the email address according to RFC 7817 [Mel16] when sending emails.

6.3.2.1 Initially Compromised Nodes

The attacker starts with a set of *nodes* considered compromised initially, namely ASes, IPs and domains located in the attacking country.

$$\frac{x \in \text{AS} \cup \text{IP} \cup \text{Dom} \quad cn \in \text{Cntry} \quad x \xrightarrow{\text{LOC}} n \quad \mathsf{C}(cn)}{\mathsf{C}(x)}$$

An IP is also considered compromised if it belongs to an attacking country or a compromised AS. If a domain resolves to such an IP, this domain is considered compromised, too. Vice versa, if a compromised domain resolves to an IP, the IP is considered compromised as well.

$$\frac{i{\in}\text{IP} \quad a{\in}\text{AS} \quad i \xrightarrow{\text{ORIG}} a \quad \mathsf{C}(a)}{\mathsf{C}(i)} \quad \frac{d{\in}\text{Dom} \quad i{\in}\text{IP} \quad d \xrightarrow{\text{A}} i \quad \mathsf{C}(i)}{\mathsf{C}(d)} \quad \frac{d{\in}\text{Dom} \quad i{\in}\text{IP} \quad d \xrightarrow{\text{A}} i \quad \mathsf{C}(d)}{\mathsf{C}(i)}$$

Whether or not a server is part of a given jurisdiction is an ongoing debate [Tri16], and subject to the attacker's legal system. We sidestep all these juridical nuances; policy-makers should explicitly state all compromised domains, ASes and IPs based on an actual legal assessment. Whether or not a server is part of a given jurisdiction is subject to the attacker's legal system and out of the scope of this work; we sidestep this question by assuming jurisdiction to equal geographical location. Policy-makers should explicitly state all compromised domains, ASes, and IPs based on an actual legal assessment.

6.3.2.2 Attacks via Routing

Next, we consider routing attacks at the AS level. Each IP is part of an AS. An IP packet may traverse several AS on the way from source AS to target AS. If *any* of these is compromised, this route is considered compromised. Note that complete information about the set of routes a packet takes depends on private contracts between ASes implemented via custom routing policies. Furthermore, the routing changes dynamically, e.g, when contracts foresee quotas. We approximate these routes using measurements from the RIPE database (see Section 6.6.4).

$$\frac{\begin{array}{cccc} i,j\in\mathrm{IP} & a,b,c\in\mathrm{AS} & \mathsf{C}(b) & \mathsf{nVPN}(a,c) \\ i\xrightarrow{\mathrm{ORIG}}a & j\xrightarrow{\mathrm{ORIG}}c & \multicolumn{2}{c}{a\xrightarrow{\mathrm{RTE}(b)}c} \end{array}}{\mathsf{I}^{\mathsf{R}}(i,j)}$$

Routing level mitigations presented in the next section consider securing the entire communication between two ASes a and c using IPsec. The *absence* of this mitigation is represented via a predicate $\mathsf{nVPN}(a,c)$, hence, if this predicate is true, then the integrity of the communication from $i \in \mathrm{IP}$ to $j \in \mathrm{IP}$ must be considered compromised, if there is some compromised AS b on some (possibly multi-hop) route from a to c.

6.3.2.3 Integrity of Domain/MX Resolution

A domain's resolver (i.e., the last recursive name server, c.f. Section 6.6) performs iterative domain resolution whenever it cannot respond with a cached DNS record. To resolve, e.g., www.google.com.[2], one of the root servers, which are authoritative for ., is queried and returns the authoritative name server for com, which is then queried and returns the authoritative name server for google.com, which finally responds to the query by providing the IP of www.google.com. Consequently, if *any* requested name server is under adversarial control, the attacker can alter the requested DNS record in whatever way he or she likes. Nevertheless we restrict the adversary to not compromise the root servers' responses, as such an attack would put the DNS infrastructure as a whole into question, and we assume the adversary is sneaky. Let $d \xrightarrow{\mathrm{DNS}} e$ be given for any name server d that could be queried during resolution of domain e. This would include all root servers, no matter which domain e is chosen. However, we explicitly exclude those, as such an attack would put the DNS infrastructure as a whole into question, and we assume the adversary is sneaky. If there is a compromised domain potentially queried during resolution, the whole resolution is seen as compromised.

$$\frac{d,e\in\mathrm{Dom}\quad i\in\mathrm{IP}\quad d\xrightarrow{\mathrm{DNS}}e\quad e\xrightarrow{\mathrm{A}}i\quad \mathsf{C}(i)}{\mathsf{I}^{\mathsf{DNS}}(d)}\qquad (r_{dns-ns})$$

If the resolver itself is already under control of the attacker, then every resolution via this resolver must be seen as compromised.

$$\frac{d,e\in\mathrm{Dom}\quad i\in\mathrm{IP}\quad d\xrightarrow{\mathrm{RSLVR}}i\quad \mathsf{C}(i)}{\mathsf{I}^{\mathsf{DNS}}(d,e)}$$

[2] The terminating '.' is typically omitted by convention.

We also model that the name resolution itself is susceptible to routing attacks, e.g., via packet injection, and maintain another predicate, $\mathsf{I}^{\mathsf{DNS}}(d, e)$, formalizing a lack of integrity of the domain resolution of e *from the perspective of a domain* d in case that the resolution d performs can be compromised via packet routing. In Section 6.4, we will consider DNSSEC, which mitigates DNS spoofing via packet injection but not DNS spoofing via compromised authoritative name servers or a compromised resolver[3]. Hence we add the precondition $\mathsf{nDNSSEC}(e)$ for e the domain to be resolved.

$$\frac{d,e,f{\in}\mathrm{Dom} \quad d\xrightarrow{\mathrm{RSLVR}}r \quad \mathsf{nDNSSEC}(e) \qquad (e\xrightarrow{\mathrm{DNS}}f{\wedge}f\xrightarrow{A}i{\wedge}\mathsf{I}^{\mathsf{R}}(r,i)){\vee}(d\xrightarrow{A}i{\wedge}\mathsf{I}^{\mathsf{R}}(i,r))}{\mathsf{I}^{\mathsf{DNS}}(d,e)}$$

(Remark: we use a disjunction in the precondition for brevity. This is to be read as two rules, one for each disjunct.)

6.3.2.4 Confidentiality

The remaining attacker actions translate initial compromise and attacks on the routing and the name resolution to a loss of confidentiality on the protocol level. A peculiarity of the SMTP protocol is that emails are not delivered to the domain indicated by the email, but to the MX entry for this domain, i.e., a domain, which in turn is resolved to an IP[Kle08]. This dependence of application-level features from name resolution has undesired effects on certificate validation, as we will see.

If an email server is initially compromised, all connections to or from it are considered unconfidential immediately.

$$\frac{d,e{\in}\mathsf{Provider} \quad i,j{\in}\mathrm{IP} \quad d\xrightarrow{\mathrm{MX}}d' \quad e\xrightarrow{\mathrm{MX}}e' \qquad d'\xrightarrow{A}i \quad e'\xrightarrow{A}j \quad \mathsf{C}(i){\vee}\mathsf{C}(j)}{\mathsf{unconf}(d,e)}$$

If the integrity of the name resolution of an MX is compromised, TLS may preserve the confidentiality of communication by validating the certificate of the recipient. However, this requires the recipient to support TLS (which was the case for all email providers we considered), and the sender to terminate TLS connections if the recipient's TLS certificate does not validate for the MX's host name (which rarely happens). It is a known weakness of the STARTTLS mechanism for securing SMTP connections that an active adversary can intercept and silently drop the attempt to elevate the connection to STARTTLS, in which case its fallback mechanism to unencrypted SMTP results in a loss of confidentiality. Despite being an active attack, it scales well and is in fact used for mass surveillance, e.g., in Thailand [Blu17]. Similarly, SMTP over TLS (SMTPS) remains insecure if the sender falls back to unencrypted SMTP if the connection to the receiver on port 465 fails. Consequently, in our model, both approaches are equivalent to no encryption at all.

[3] We assume that the domain's resolver is validating and that its validation is trusted in iterative resolution. This is the most common setup.

All email providers we considered do not enforce TLS connections in general, and accept even self-signed certificates. Hence, they are susceptible to this attack. This is not surprising, considering that the original STARTTLS proposal considered domain validation a 'local matter' [Hof02]. Facebook and Google are aware of this problem, and are trying to push the use of TLS for emails by investigating deployment obstacles and, to some extent, by public shaming [Inc17; Adk14a], Still, even some popular domains fail strict certificate validation [Adk14b].

If the sender does not enforce strict host validation, e.g., he or she uses optimistic STARTTLS, the attacker considered in our threat model can change a provider's MX record to point to a domain under their control, point the domain of the MX to an IP of their choice, or intercept packets on the route between their respective email servers.

$$\frac{\begin{array}{c} d,e{\in}\mathsf{Provider} \quad d{\neq}e \quad d\xrightarrow{\text{MX}}d' \\ \mathsf{nTLS}^{\mathsf{snd}}(d) \quad \mathsf{I}^{\mathsf{DNS}}(e){\vee}\mathsf{I}^{\mathsf{DNS}}(d',e) \end{array}}{\mathsf{unconf}(d,e)}$$

The predicate $\mathsf{nTLS}^{\mathsf{snd}}(d)$, indicating that d does not enforce strict host validation when sending emails, is initially always false – currently no email provider does this, the original STARTTLS proposal even considered domain validation a 'local matter' [Hof02]. Still, some popular domains fail strict certificate validation [Adk14b]. We will consider the cost of changing the status quo in Section 6.4. Consequently, the attacker can modify the IP to which the MX entry (which is a domain) itself is resolved.

$$\frac{\begin{array}{c} d,e{\in}\mathsf{Provider} \quad d{\neq}e \quad d\xrightarrow{\text{MX}}d' \quad e\xrightarrow{\text{MX}}e' \\ \mathsf{I}^{\mathsf{DNS}}(e'){\vee}\mathsf{I}^{\mathsf{DNS}}(d',e') \quad \mathsf{nTLS}^{\mathsf{snd}}(d) \end{array}}{\mathsf{unconf}(d,e)}$$

Routing attacks also apply to TLS but can be countered by proper certificate validation, and also by using DANE or SMTP STS (cf. Section 6.4).

$$\frac{\begin{array}{c} d,e{\in}\mathsf{Provider} \quad d{\neq}e \quad i,j{\in}\text{IP} \quad d\xrightarrow{\text{MX}}d' \quad e\xrightarrow{\text{MX}}e' \\ d'\xrightarrow{A}i \quad e'\xrightarrow{A}j \quad \mathsf{I}^{\mathsf{R}}(i,j) \quad \mathsf{nTLS}^{\mathsf{snd}}(d) \quad \mathsf{nDANE}^{\mathsf{rcv}}(e) \end{array}}{\mathsf{unconf}(d,e)}$$

Even enforcing strict domain validation allows an attacker to intercept emails, as the integrity of the domain name resolution procedure is required to identify the MX responsible for an email domain. Most client certificates contain the domain name of this MX server, but not the email domain. Consequently, a connection between two email domains can still remain unconfidential, even if TLS is enforced, if the adversary is able to refer communication to another server by means of the MX entry. In 2016, the server identity check procedure was updated with RFC 7817: a valid certificate now needs to contain the email domain (domain id, the last part of the email address), as well as the MX domain [Mel16]. This prevents the above attack, hence the attacker action in our model only applies if the email provider performs no email domain verification according to RFC 7817 ($\mathsf{nRFC7817}(d)$). None of the email providers we encountered enforced this strict validation. hence the predicate $\mathsf{nRFC7817}(d)$ will only be removed as a result of a possible defender action (see next section).

$$\frac{\begin{array}{c} d{,}e{\in}\mathsf{Provider} \quad d{\neq}e \quad d\xrightarrow{\text{MX}}d' \\ \mathsf{I}^{\mathsf{DNS}}(e){\vee}\mathsf{I}^{\mathsf{DNS}}(d'{,}e) \quad \mathsf{nRFC7817}(d) \end{array}}{\mathsf{unconf}(d, e)}$$

6.4 Defender Model

We describe the defender model in terms of defender actions that can be composed to mitigation strategies to minimize the attacker's objective value $\mathcal{R}$. In contrast to attacker actions, each defender action f is associated with a cost $c(f) \in \mathbb{R}_0^+$ (the cost from the defender's point of view). We estimate this cost based on publicly available data (our analysis algorithm performs well, cf. Section 6.7; state actors with access to detailed information will be able to refine our cost estimates and obtain more accurate results). The complete set of defender actions, including short descriptions, can be found in Section 6.5.2.

6.4.1 Provenance and Role of the Defender

We take the view of a state actor adhering to the rule of law, trying to estimate the cost mitigations would impose *on the companies affected*. We assume all companies providing services to users from the defending country are under this country's jurisdiction. Depending on the mitigation, and the existence or absence of regulatory entities within this country, mitigations could be implemented by 1) recommendations coming from regulatory bodies or government agencies, 2) regulatory standards enforced by such bodies, or 3) laws permitting direct intervention in case of national security risks. The actual cost of enforcing these mitigations are not only of monetary, but also of political nature, including potential insecurity in long-term investments, additional administrative costs, necessary political cost and, most importantly, eventual loss of individual and entrepreneurial freedom. We concentrate on the economic cost to provide a result with a clear interpretation.

6.4.2 Routing Mitigations

Routing-level attacks affect both communication and name resolution. We thus consider securing packets routed between two ASes via IPSEC [KS05; Hou05]. We add a rule permitting the defender to set the predicate $\mathsf{nVPN}(a, b)$ to false, for any two ASes located on its soil, which prevents routing attacks as discussed in Section 6.3.2.2.

$$\frac{a, b \in \text{AS} \quad a, b \xrightarrow{\text{LOC}} cn_D}{\neg\mathsf{nVPN}(a, b)} \; c = \$56\,000$$

Cost Estimation

Public data on AS-level throughput is incomplete, but the data we have suggest that the vast majority of ASes are connected with a link speed of 10Gbit/s, hence we estimate the cost of providing this throughput at peak times [Pee17]. The computational requirements are well understood, two dedicated routers (one for each side) for \$24 000 each are sufficient [RGE$^+$16]. Adding about 80 consulting hours for configuration and testing [Cha17], and annual cost for support and maintenance [Cha17], we arrive at an estimate of $80 \cdot \$100 + \$48\,000 = \$56\,000$ for the cost of deploying a VPN connection. We neglect the energy cost, as an existing router would likely be replaced, and the additional computational cost is almost exclusively symmetric cryptography, which is very efficient on these devices.

6.4.3 DNS-Level Mitigations

We consider two countermeasures against attacks to domain resolution: DNSSEC and DANE. DNSSEC was designed to provide, among other properties, origin authentication and data integrity for DNS data [AAL$^+$05a; R A05; AAL$^+$05b]. Let $d \xrightarrow{\text{NS}} e$ be given for any name server e that is authoritative for d. DNSSEC mitigates DNS-level attacks that rely on intercepting packets between an email provider and its resolver or between the resolver and an authoritative name server. It does not provide protection against authoritative name servers under adversarial control. As deployment of DNSSEC on the root servers and the defender countries' TLDs is completed, this mitigation is available as of now. We consider the employment cost per company, hence all servers for which a provider's name server is authoritative are mitigated in one step. As a side effect, email providers which are part of the same company share this cost if the same name server is authoritative for them. Some email providers, such as web.de and GMX in Germany, have already deployed it.

$$\frac{\substack{f \in \mathsf{Provider} \\ \{d_1,\ldots,d_l\}=\{d\in\mathrm{Dom} \mid d \xrightarrow{\text{DNS}} e \wedge f \xrightarrow{\text{NS}} e\}}}{\neg\mathsf{nDNSSEC}(d_1) \wedge \cdots \wedge \neg\mathsf{nDNSSEC}(d_l)} \; c = \$366\,342$$

DNSSEC by itself is only useful against packet injection attacks on resolution, as any name server consulted during resolution of a domain is part of the trust chain. But the integrity it provides can be used to indicate that the communication partner prefers communication via TLS. This sidesteps the deployment issue that strict certificate validation in TLS faces. There are two technologies which can be used to this end, DANE (DNS-Based Authentication of Named Entities [DH15]) and SMTP STS (Strict Transport Security [MRL$^+$16]). Both have different aims and approaches: SMTP STS allows email transfer agents (MTA) to publish their intent to use TLS, e.g., via text records in DNS, while DANE first and foremost transmits information about which certificates should be accepted, e.g., by publishing the hash of the end-to-end certificate. Thus DANE may be used to indicate that TLS shall be used for email communication by providing the certificate's hash. While DANE can be used to take over some tasks of the public-key infrastructure (PKI), in our model the PKI is fully trusted. Hence with respect to our threat model, both technologies are providing essentially the same functionality.

$$\frac{\overset{f \in \mathsf{Provider}}{f \xrightarrow{\text{NS}} e \quad d \xrightarrow{\text{DNS}} e \quad \neg \mathsf{nDNSSEC}(d)}}{\neg \mathsf{nDANE}^{\mathsf{rcv}}(f)} \; c = \$4\,000$$

Cost Estimation

We assume all email providers have a DNSSEC-aware resolver, as software is readily available to this end, and consider only the cost at the receiver's side, where the DNS infrastructure needs to be upgraded. We calculated the cost for deploying DNSSEC based on an extensive survey [NA10] among 19 companies about their capital expenditures (CAPEX) for upgrading their infrastructure to handle DNSSEC. Email providers are zone operators, judging from the size of the companies that took part, it seemed appropriate to the take the maximum CAPEX, which is €335 590 ($366 342).[4] This number might be surprisingly high, but DNSSEC comes with considerable operational efforts witnessed, e.g., by the failed key rollover of the .gov TLD in 2013.

The deployment itself is not a complex task; planning and implementing should not need more than 40 working hours. Therefore, we fix the cost for reconfiguration to be $\mathit{cost} = 40 * \$100 = \$4\,000$. Our model could be used for a more nuanced treatment distinguishing the two approaches; yet whichever has the lowest cost would dominate the other, as we presume the PKI to be trusted. In other words, some advantages of DANE over SMTP STS simply do not apply to our threat model.

6.4.4 Mitigation by Adoption of Secure Standards

As discussed in Section 6.3, the connection between two email providers can be secured by using TLS, however, the efficacy hinges on the way domain validation is handled. All large email providers employ opportunistic SMTPS or STARTTLS, hence for an active adversary, using TLS without any domain validation is immediately vulnerable to packet injection and DNS spoofing attacks. Hence in our model, this mitigation would not make any difference. The same holds for the adoption of STARTTLS, which is also known to be ineffective against active adversaries. We thus consider the adoption of SMTPS with 1) validation of the server's domain, and 2) validation of both the server's domain and the domain part of the email address, i.e., according to RFC 7817.

$$\frac{d \in \mathsf{Provider}}{\neg \mathsf{nTLS}(d)} \; c = \$4k \qquad \frac{d \in \mathsf{Provider}}{\neg \mathsf{nTLS}(d) \land \neg \mathsf{nRFC7817}(d)} \; c = \$4\,080$$

Cost Estimation

Similar to the adoption of DANE or SMTP TLS, strict enforcement of TLS should not need more than 40 working hours by external consultants, i.e., $40 * 100 = \$4\,000$. The additional running cost imposed by cryptography is negligible. First, the additional costs for the record protocol are very cheap [Gri13], second, modern email transfer agents support connection sharing, which is very efficient in our setting where a small

[4] The survey also revealed that some organizations had only small upgrade costs, because of overcapacities in CPU and storage. By taking the maximal cost we assume the worst-case, where an infrastructure upgrade is necessary.

number of popular email providers dominate the market. Third, as about 80% to 90% of communication enjoys (opportunistic) TLS already, a relatively small amount of traffic is affected. The implementation cost of RFC 7817 is the same as for strict certificate validation, but the certificate format needs to be adapted to this recent standard, which amounts to about $80 for the first year. This is only the cost of implementation, which we can get reliable estimates for. Our results (cf. scenario S2 in Section 6.7.2) indicate that strict certificate validation would have been widely deployed if these were the only cost imposed by deployment. However, unless all legitimate email providers have valid certificates, strict validation will come at a cost of functionality. Users will not be able to reach other users on servers with invalid certificates. We will thus also discuss a model reflecting the loss of revenue due to users quitting the provider in frustration.

6.4.5 Mitigation by Relocation

If the adversary controls infrastructure used by the defender, e.g., the email server itself, the only remedy is to set up trustworthy infrastructure or relocate it to put it under a different jurisdiction.

$$\frac{e \in \mathsf{Dom} \quad \{d_1, \cdots, d_l\} = \{d \mid d \xrightarrow{\text{DNS}} e\}}{d_1 \not\xrightarrow{\text{DNS}} e \wedge \cdots \wedge d_l \not\xrightarrow{\text{DNS}} e} \quad c = \$10\,000$$

$$\frac{d \in \mathsf{Provider} \quad \{e_1, \cdots, e_l\} = \{e \mid d \xrightarrow{\text{MX}} e\}}{d \not\xrightarrow{\text{MX}} e_1 \wedge \cdots \wedge d \not\xrightarrow{\text{MX}} e_l} \quad c = \$1 \cdot u(d)$$

Although politically difficult, it is not completely unlikely in the context of European privacy law. In October 2015, following a court decision by the Court of Justice of the European Union, the safe harbor agreement between the EU and US was declared invalid on the grounds that the US was not supplying an equally adequate level of protection against surveillance for data being transferred there. A new agreement, the EU-US Privacy shield, was quickly put in place after this decision, however, the European Data Protection Supervisor stated in May 2016 that 'the Privacy Shield, as it stands, is not robust enough to withstand future legal scrutiny before the [European] Court' [Eur16]. We are excluding authoritative name servers for top-level domains (TLDs) from relocation, but name servers below can be relocated, removing all dependencies to the compromised TLD.

Cost Estimation

While relocation cost depends a lot on a company's location, we aim for a uniform treatment which only takes into account the number of users, $u(d)$, a provider d has, ignoring the difference in cost of running the servers at the new location, e.g., rent, electricity, insurance, tax, etc.

Planning and executing the relocation amounts to ca. $10,000 per rack [Inf17]. This gives the cost per name server relocation. For email servers, the relocation costs scale with the users, hence we multiply these cost with the number of users, as one rack can serve 10k users [Sop17]. As relocation costs are similar to moving the same infrastructure to a (trusted) cloud service [Shi14], we do not distinguish the two scenarios.

6.5 Detailed Email Threat and Defender Model

We detail the complete threat and defender model described in Section 6.3 and Section 6.4 here. If you want to skip this, you can continue reading Section 6.6 on page 96.

For brevity, we will use disjunctions in the precondition of rules as a shorthand for two rules, one for each disjunct. In logical terms, an action is given by a *precondition pre*, a boolean formula over proposition literals, and a *postcondition post*, a conjunction over proposition literals, and is written as

$$\frac{pre}{post.}$$

We will give each rule, followed by the intuition of what kind of attack it represents.

6.5.1 Complete Threat Model

6.5.1.1 Initially Compromised Nodes

Rule $r_{init-loc}$:

$$\frac{x \in \mathrm{AS} \cup \mathrm{IP} \cup \mathrm{Dom} \quad cn \in \mathrm{Cntry} \quad x \xrightarrow{\mathrm{LOC}} n \quad \mathsf{C}(cn)}{\mathsf{C}(x)}$$

Intuition All autonomous systems, IPs and domains associated to the attacking country are initially under control of the attacker.

Rule $r_{init-as}$:

$$\frac{i \in \mathrm{IP}\ a \in \mathrm{AS}\ i \xrightarrow{\mathrm{ORIG}} a\ \mathsf{C}(a)}{\mathsf{C}(i)}$$

Intuition If an AS is under the control of the attacker, any IP which is part of the AS is also under control of the attacker.

Rule $r_{init-dom}$:

$$\frac{d \in \mathrm{Dom}\ i \in \mathrm{IP}\ d \xrightarrow{\mathrm{A}} i\ \mathsf{C}(i)}{\mathsf{C}(d)}$$

Intuition If an IP is under the control of the attacker, any domain that resolves to it (even if the attacker cannot interfere with the resolution), is also under the control of the attacker.

Rule $r_{init-ip}$:

$$\frac{d \in \text{Dom}\ i \in \text{IP}\ d \xrightarrow{\text{A}} i\ \mathsf{C}(d)}{\mathsf{C}(i)}$$

Intuition If a domain is under the control of the attacker, any IP it resolves to (even if the attacker cannot interfere with the resolution) is also under the control of the attacker.

6.5.1.2 Attacks via Routing

Rule $r_{injection}$:

$$\frac{\substack{i,j \in \text{IP} \quad a,b,c \in \text{AS} \quad \mathsf{C}(b) \quad \mathsf{nVPN}(a,c) \\ i \xrightarrow{\text{ORIG}} a \quad j \xrightarrow{\text{ORIG}} c \quad a \xrightarrow{\text{RTE}(b)} c}}{\mathsf{I}^{\mathsf{R}}(i,j)}$$

Intuition If the attacker controls an AS b which transfers packets from an IP i belonging to AS a to some IP address j belonging to AS c and this particular connection is not secured via the VPN mitigation, we assume that the integrity of the communication from i to j is compromised.

On the resolution and application level, we are only concerned with communication between IPs, the domains resolve to. Thus, this rule covers all relevant routing attacks.

6.5.1.3 Integrity of Domain/MX Resolution

Rule r_{dns-ns}:

$$\frac{d,e \in \text{Dom} \quad i \in \text{IP} \quad d \xrightarrow{\text{DNS}} e \quad e \xrightarrow{\text{A}} i \quad \mathsf{C}(i)}{\mathsf{I}^{\mathsf{DNS}}(d)} \tag{r_{dns-ns}}$$

Intuition If the attacker controls any name server that could be queried during resolution, we consider the integrity of the domain name resolution compromised.

Although unspecified by RFC 3207, name servers commonly attach an A record whenever they respond to a resolution request with an DNS entry pointing to another name server [Hof02]. This speeds up the resolution and has the pleasant side effect that the integrity of the resolution does not depend on whether these authoritative the integrity of the resolution of the domain names of the name servers requested, hence the transitive rule for $\mathsf{I}^{\mathsf{DNS}}$ is not at the attacker's disposal.

Rule $r_{dns-res}$:

$$\frac{d,e \in \text{Dom} \quad i \in \text{IP} \quad d \xrightarrow{\text{RSLVR}} i \quad \mathsf{C}(i)}{\mathsf{I}^{\mathsf{DNS}}(d,e)}$$

Intuition If the attacker controls the resolver of a domain, we consider the integrity of any domain name resolution this domain attempts compromised. (Technically, r is an IP address, but we simplified this and the following rule for presentation.)

Rule $r_{dns-route-res}$:

$$\frac{d,e \in \mathrm{Dom} \quad i \in \mathrm{IP} \quad d \xrightarrow{\mathrm{RSLVR}} r \quad d \xrightarrow{A} i \quad \mathsf{I}^{\mathsf{R}}(i,r) \quad \mathsf{nDNSSEC}(e)}{\mathsf{I}^{\mathsf{DNS}}(d,e)}$$

Intuition If the attacker controls the route from a domain to the resolver this domain uses, we consider the integrity of any domain name resolution this domain attempts compromised, unless the integrity of the resolution is guaranteed by DNSSEC. Dax and Künnemann [DK21a] investigated the symbolic soundness of the threat model presented here and noted that DNSSEC would currently not be an effective mitigation against domain poisoning attacks mounted between the local and the ISP's resolver. The reason is that we assume the local resolvers to validate signatures themselves which is, as of right now, not the case. We still make the unrealistic assumption here to investigate the potential effectiveness of DNSSEC as soon as the local resolvers implement it.

Rule $r_{dns-route-ns}$:

$$\frac{\begin{array}{c} d,e,f \in \mathrm{Dom} \quad r \in \mathrm{IP} \quad d \xrightarrow{\mathrm{RSLVR}} r \quad e \xrightarrow{\mathrm{DNS}} f \\ f \xrightarrow{A} i \quad \mathsf{I}^{\mathsf{R}}(r,i) \quad \mathsf{nDNSSEC}(e) \end{array}}{\mathsf{I}^{\mathsf{DNS}}(d,e)}$$

Intuition If the attacker controls the route from a resolver to some authoritative name server potentially queried during resolution, we consider the integrity of the resolution for this domain name compromised, unless the integrity of the resolution is guaranteed by DNSSEC.

6.5.1.4 Confidentiality

Rule $r_{compromise}$:

$$\frac{\begin{array}{c} d,e \in \mathsf{Provider} \quad i,j \in \mathrm{IP} \quad d \xrightarrow{\mathrm{MX}} d' \quad e \xrightarrow{\mathrm{MX}} e' \\ d' \xrightarrow{A} i \quad e' \xrightarrow{A} j \quad \mathsf{C}(i) \lor \mathsf{C}(j) \end{array}}{\mathsf{unconf}(d,e)}$$

Intuition If an email server is already compromised, e.g., if it is hosted by an adversarial country, the attacker can compromise the confidentiality of the communication between two email providers.

Rule $r_{fake-mx}$:

$$\frac{\begin{array}{c} d,e \in \mathsf{Provider} \quad d \neq e \quad d \xrightarrow{\mathrm{MX}} d' \\ \mathsf{nTLS}^{\mathsf{snd}}(d) \quad \mathsf{I}^{\mathsf{DNS}}(e) \lor \mathsf{I}^{\mathsf{DNS}}(d',e) \end{array}}{\mathsf{unconf}(d,e)}$$

Intuition If the sender does not enforce strict host validation, e.g., by using optimistic STARTTLS, the attacker can compromise the confidentiality of the communication between two email providers by changing a provider's MX record to point to a domain under their control.

Rule $r_{fake-ip}$:

$$\frac{d{,}e{\in}\mathsf{Provider} \quad d{\neq}e \quad d\xrightarrow{\text{MX}}d' \quad e\xrightarrow{\text{MX}}e' \\ \mathsf{I}^{\mathsf{DNS}}(e'){\vee}\mathsf{I}^{\mathsf{DNS}}(d'{,}e') \quad \mathsf{nTLS}^{\mathsf{snd}}(d)}{\mathsf{unconf}(d,e)}$$

Intuition If the sender does not enforce strict host validation, e.g., by using optimistic STARTTLS, the attacker can compromise the confidentiality of the communication between two email providers by pointing the domain of the MX to an IP of their choice.

Rule $r_{intercept}$:

$$\frac{d{,}e{\in}\mathsf{Provider} \quad d{\neq}e \quad i{,}j{\in}\text{IP} \quad d\xrightarrow{\text{MX}}d' \quad e\xrightarrow{\text{MX}}e' \\ d'\xrightarrow{A}i \quad e'\xrightarrow{A}j \quad \mathsf{I}^{\mathsf{R}}(i{,}j) \quad \mathsf{nTLS}^{\mathsf{snd}}(d) \quad \mathsf{nDANE}^{\mathsf{rcv}}(e)}{\mathsf{unconf}(d,e)}$$

Intuition If the sender does not enforce strict host validation, e.g., he or she is using optimistic STARTTLS, and DANE is not deployed, the attacker can compromise the confidentiality of the communication between two email providers by intercepting packets on the route between their respective email servers.

Rule $r_{fake-mx-strict}$:

$$\frac{d{,}e{\in}\mathsf{Provider} \quad d{\neq}e \quad d\xrightarrow{\text{MX}}d' \\ \mathsf{I}^{\mathsf{DNS}}(e){\vee}\mathsf{I}^{\mathsf{DNS}}(d'{,}e) \quad \mathsf{nRFC7817}(d)}{\mathsf{unconf}(d,e)}$$

Intuition If the sender does not enforce certificate validation according to RFC 7817, e.g., by using optimistic STARTTLS or strict validation on the hostname only, the attacker can compromise the confidentiality of the communication between two email providers by changing a provider's MX record to point to a domain under their control.

6.5.2 Complete Defender Model

6.5.2.1 Routing Mitigation

$$\frac{a,b \in \text{AS} \quad a,b \xrightarrow{\text{LOC}} cn_D}{\neg\mathsf{nVPN}(a,b)} \quad c = \$56\,000$$

Intuition The connection between any two AS, adjacent or not, can be secured by deploying a VPN, provided the AS is under the jurisdiction of the defending country cn_D.

6.5.2.2 DNS-Level Mitigations

$$\frac{f{\in}\mathsf{Provider} \\ \{d_1{,}...{,}d_l\}{=}\{d{\in}\text{Dom}|d\xrightarrow{\text{DNS}}e{\wedge}f\xrightarrow{\text{NS}}e\}}{\neg\mathsf{nDNSSEC}(d_1) \wedge \cdots \wedge \neg\mathsf{nDNSSEC}(d_l)} \quad c = \$366\,342$$

Intuition An email provider can deploy DNSSEC on all domains below any authoritative name server for the domain that identifies them for a fixed price. Below is interpreted with respect to the DNS hierarchy, every domain whose resolution depends on such an authoritative name server is considered ‘below‘.

$$\frac{f \in \mathsf{Provider} \quad d \xrightarrow{\text{DNS}} e \quad f \xrightarrow{\text{NS}} e \quad \neg\mathsf{nDNSSEC}(d)}{\neg\mathsf{nDANE}^{\mathsf{rcv}}(f)} \; c = \$4\,000$$

Intuition For all domains below any authoritative name server for a domain that identifies a provider, if this domain supports DNSSEC already, it can also deploy DANE/SMTP STS.

6.5.2.3 Mitigation by Adoption of Secure Standards

$$\frac{d \in \mathsf{Provider}}{\neg\mathsf{nTLS}(d)} \; c = \$4\,000$$

Intuition Spending cost for TLS with strict certificate validation.

$$\frac{d \in \mathsf{Provider}}{\neg\mathsf{nTLS}(d) \wedge \neg\mathsf{nRFC7817}(d)} \; c = \$4\,080$$

Intuition Spending cost for TLS with strict certificate validation according to RFC 7817, additional cost for new certificates.

6.5.2.4 Mitigation by Relocation

$$\frac{e \in \mathrm{Dom} \quad \{d_1, \cdots, d_l\} = \{d \mid d \xrightarrow{\text{DNS}} e\}}{d_1 \not\xrightarrow{\text{DNS}} e \wedge \cdots \wedge d_l \not\xrightarrow{\text{DNS}} e} \; c = \$10\,000$$

Intuition For a fixed price, a name server can be relocated, removing all dependencies on this name server.

$$\frac{d \in \mathsf{Provider} \quad \{e_1, \cdots, e_l\} = \{e \mid d \xrightarrow{\text{MX}} e\}}{d \not\xrightarrow{\text{MX}} e_1 \wedge \cdots \wedge d \not\xrightarrow{\text{MX}} e_l} \; c = \$1 \cdot u(d)$$

Intuition For a price that scales with the number of users, an email server can be relocated, removing all dependencies on this email server.

6.6 Data Acquisition

In order to evaluate our mitigation analysis, we select attacker and defender countries we consider relevant. We use market share data to determine the most popular email service providers in each defender country. Then we acquire DNS and routing data to derive the property graph described in Section 6.2.3 and instantiate the threat model.

6.6.1 Attacker Countries

We have to choose which country is taking the ‘attacker’ role, but stress that this word is to be interpreted in the information security sense. We chose the following criteria to avoid politicizing this decision: As we are interested in countries which are both capable and likely to engage in large-scale email sniffing, we consider the purported spending on intelligence services [Hip09], military spending [Ins16], and the press freedom index [Bor16], the first two as indicators for the intelligence capabilities, and the third for the plausibility of such an endeavor.

We consider the seven countries which are both in the top 10 w.r.t. spending on intelligence and military and, in addition, in the bottom 140 of the press freedom index. We also consider the so-called Five Eyes agreement (Australia, Canada, New Zealand, the United Kingdom and the United States) as this alliance is known to engage in email-sniffing [Nor14] . According to a document leaked by Edward Snowden, a larger group of countries is officially known as SIGINT Seniors Europe, or ‘SSEUR’ [Ren13]. This alliance, consisting of the Five Eyes plus Denmark, France, the Netherlands, Norway, Germany, Belgium, Italy, Spain and Sweden is also known as ‘Fourteen Eyes’. We thus consider the United States, Japan, China, Russia, Italy, Mexico, South Korea, having the role of the attacker, as well as Five Eyes and Fourteen Eyes.

6.6.2 Defender Countries & Vantage Points

We chose defender countries according to the number of Internet users in the country. Table 6.3 shows the nine countries with the most Internet users (in absolute numbers). The Internet as a whole, and domain resolution in particular, looks different when observed from different countries, e.g., due to CDNs (e.g., DNS-based request routing [BCN$^+$03] [WJR$^+$15]), Anycast DNS [AL06] and censorship. Unfortunately, we could not get access to machines in certain countries, therefore we chose to omit them. This resulted in the selection of: China, USA, Brazil, Russia and Germany. At each of those vantage points, we instantiate a new property graph with the data gathered from the respective server. We, furthermore, obtained market share data to evaluate attack impacts and identify the most popular email service providers in the defender countries, see, e.g., Figure 6.3 or Figure 6.8 - Figure 6.11 for the other countries.

6.6.3 DNS-Related Data

Our data acquisition starts with DNS queries for collecting the DNS-related information. DNS is, in fact, a database that provides lookup information for different relationships, including domain name to IP address, email address to email servers, email server to an IP address and so on. In our work, we collect the DNS records relevant to our threat model, namely, A records, i.e., mapping domain name to an IP, MX records, i.e., email transfer agents used by the domain name and NS record, i.e., authoritative name servers used by that domain. We add nodes and relationships (see Table 6.1) corresponding to these records to the property graph (see Section 6.2.3). To identify the local DNS resolver (or the DNS forwarder) used by email servers for name resolution, we monitor the inbound traffic at an external name server, to identify the IP address of the resolver.

Table 6.3: Countries by the number of Internet users [Int16; Dun12].

Country	Internet users	Email	Probes	Server
China	731 434 547	87%	×	✓
India	462 124 989	68%	×	×
United States	286 942 362	88%	×	✓
Brazil	122 796 320	80%	×	✓
Japan	118 131 030	75%	×	×
Russia	105 311 724	86%	×	✓
Nigeria	86 436 611	—	×	×
Mexico	72 945 992	85%	×	×
Germany	70 675 097	89%	✓	✓

To this end, we set up an email server and added its domain name as an MX record into our DNS zone file. Next, we created email accounts with the email providers that are part of the analysis and accordingly sent emails to our email server. To avoid cached results that the local DNS resolver/forwarder might have for our domain, we change the domain name of our email server for every email provider by using a cache-busting nonce: emails are sent to resolver@x.ourdomain.com, where x is different for each email. The name server then registers which resolver requested x.ourdomain.com.

6.6.4 Routing Information and Countries

To be able to model AS-level routing attacks and their mitigation, we need routing data for the ASes where the email providers and name servers are hosted. We chose to measure actual routes, rather than simulating routing policies (e.g., Gao-Rexford) using public peering information. This improves the accuracy of our routing data, however, Germany is the only country of our selection for which we were able to get sufficiently many probes within these ASes. Our methodology can be easily adapted to the simulation approach.

For collecting routing information, we start with identifying the ASes of the email servers by querying the RIPEStat database [RIP]. We collect the information about the AS-level routing between two email servers using the RIPE Atlas [RIP17a] network of probes to acquire traceroutes between the email servers. We thus identify the actual AS-level routes package between two MTAs would take (within the time frame of our measurements). However, not all ASes host a RIPE probe, hence our analyses do not cover all possible pairs of MTAs. Table 6.3 indicates where our data was insufficient, and hence packet injection capabilities of the attacker were underestimated.

Finally, in order to establish a relationship between the servers and the countries where they belong, we add geolocation data to our property graph. To that end, we query the MaxMind [Max17] dataset and accordingly create a node of type Cntry along with an edge of type LOC that links the IP address to the selected country.

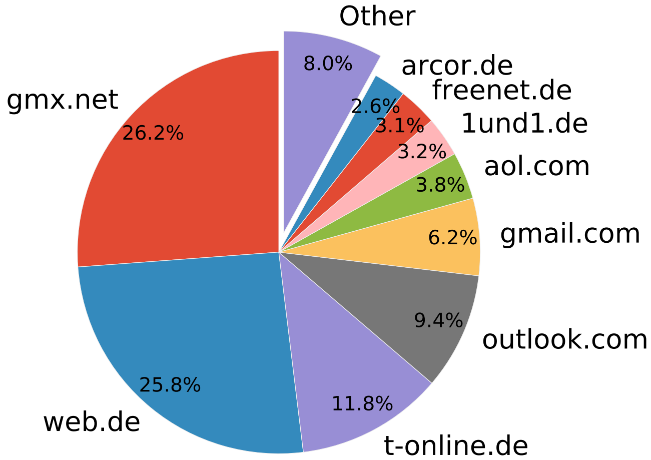


Figure 6.3: Most used (primary) email providers in Germany [GmB16].

6.7 Results and Evaluation

We now apply our methodology for the threat and defender model described in Section 6.3 and Section 6.4 to the data acquired in Section 6.6. First, we consider different cost scenarios within a case study (Five Eyes versus Germany) to discuss deployment hazards and the impact of DNSSEC. Then we discuss the results for all combinations of defenders and attackers we considered in our analysis. We visualize our results in several figures with a symlog-scaled x-axis which is linear around 0 and logarithmic o/w, the y-axis is scaled linearly. Additionally, we release our source code along with an interactive visualization of the results at [SSK^{+}17]. After pre-processing, most of the generated instances could be solved on an Intel Xeon E5-4650L machine within one minute. The largest instance required approximately 16 minutes of CPU time. The pre-processing time was dominated by IO operations, and otherwise it was negligible.

6.7.1 Threats to Validity

Provided our formal threat model is correct, we identify the following threats to validity: as mentioned before, we consider only direct monetary cost and treat defender cost uniformly regardless of country and size of company. We rely on email market share studies to estimate the impact of an attack. These consider only user's primary email addresses. The cost for TLS deployment in S3 to S6 is highly speculative. Finally, we were only able to measure routes between ASes for Germany, not for the other countries.

6.7.2 Case study: Five Eyes Versus Germany

We consider the case of the Five Eyes alliance versus Germany to discuss the effects of large-scale email sniffing on a market where foreign companies provide plenty of infrastructure (as opposed to China) but domestic companies still serve the majority of users (unlike Brazil, see Figure 6.3). Germany has a sizeable market of approx. 62.9 million email users, and it is well covered by RIPE Atlas for collecting routing

Table 6.4: Mitigation cost scenarios: all cost in \$. $c_{pess} = 0.73 \cdot 7 \cdot 0.5 = 2.555$, $c_{opt} = 0.0036 \cdot 7 \cdot 0.5 = 0.0126$.

	routing mit.	DNS mitigations		secure standards		relocation	
scenario	VPN	DNSSEC	DANE/SMTP STS	enforce TLS	RFC 7817	MX	NS
S1: unit cost	1	1	1	1	1	1	1
S2: actual spending	56 000	366 342	4 000	4 000	4 080	1/user	10k
S3: deployment (pessimistic)	56 000	366 342	4 000	4k + c_{pess}/user	(disabled)	1/user	10k
S4: deployment (opt.+pess.)	56 000	366 342	4 000	4k + c_{opt}/user	4 080+c_{pess}/user	1/user	10k
S5 (a,b): enforce TLS (opt./pess.)	56 000	(disabled)	(disabled)	4k + $c_{opt/pess}$/user	(disabled)	1/user	10k
S6 (a,b): RFC 7817 (opt./pess.)	56 000	(disabled)	(disabled)	(disabled)	4 080 + $c_{opt/pess}$/user	1/user	10k
S7: hidden attacker	(like S3, but rule r_{dns-ns} has $\mathsf{nDNSSEC}(n)$ in its premise)						

information (2114 RIPE Atlas probes, 1056 online at the time of writing). We obtained 1 332 594 routes, 176 of which are relevant to this attack scenario. There are three US companies among the top email providers in Germany; Microsoft, Google and AOL. GMX, web.de and 1&1 are part of the same company and thus share some infrastructure, e.g., they are within the same autonomous system (AS8560). These providers are also the only providers which employ DNSSEC and DANE.

Barring any mitigations, the attacker gets hold of 48 provider to provider connections, representing 45.43% of German user to user communications. Our exposition will follow the cost scenarios detailed in Table 6.4, which we will motivate as we go along.

6.7.2.1 S1: unit cost

In the unit cost scenario, each mitigation has a cost of 1, hence the defender's goal is minimizing the number of measures taken. In this first analysis (see Figure 6.4), we observe first that the attacker reward can actually be reduced to zero by enforcing TLS connections and RFC 7817 on all providers and relocating outlook.com, aol.de and gmail.com. As TLS and RFC 7817 validate the identity and the authority of the recipient MX, no relocation for name servers is necessary. Relocation is restricted to foreign email providers. All other mitigation strategies that result in zero attacker reward are dominated by this mitigation strategy, as TLS and RFC 7817 provide a cure-for-all in two steps, whereas relocation of name servers or VPNs requires mitigation at several points of the network.

6.7.2.2 S2: actual spending

Now we consider the actual cost spent on implementing these mitigations (see Figure 6.4). Here again, enforcing TLS and RFC 7817 is the prevalent solution in low-cost and high-security settings. US-based email providers are proposed to relocate their MXes. RFC 7817 is indeed more useful than the relocation of DNS servers. DNS relocation is rarely considered, as all German providers rely only on infrastructure within Germany, whereas email servers of US providers need to be relocated anyway.

It thus seems that the deployment of TLS with strict certificate validation according to RFC 7817 is the cheapest way of countering large-scale email sniffing. However, this contradicts the observation that even despite Google's and Facebook's push for encrypted server to server communication, no large email provider dares to require TLS

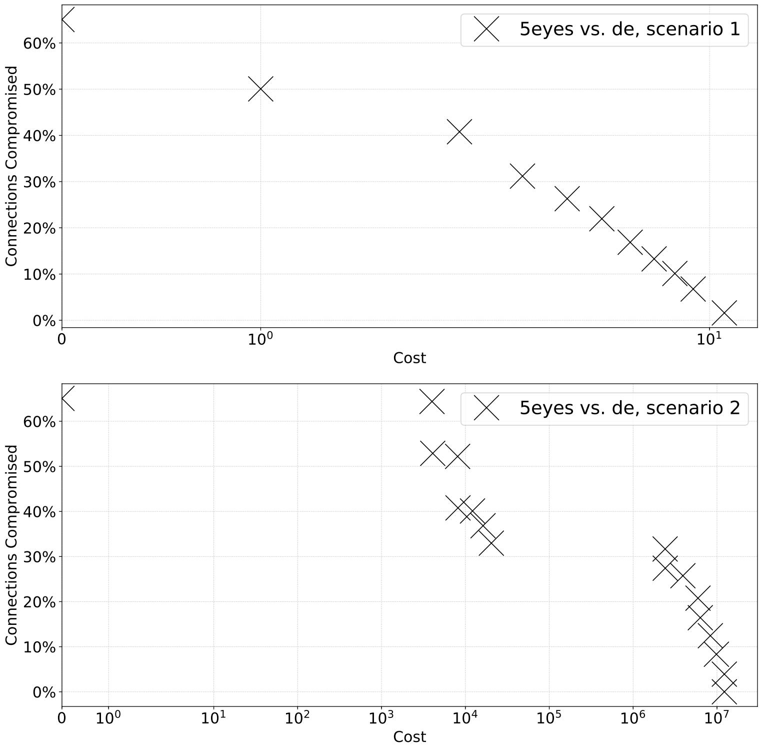


Figure 6.4: Five Eyes vs. Germany (S1) & Five Eyes vs. Germany (S2).

for SMTP connections.

6.7.2.3 S3: deployment cost, pessimistic

Opportunity cost is the main obstacle for enforcing TLS, as users leaving a service because they cannot reach their friends or business partners produces a loss in revenue.[5] We consider the average revenue per user (ARPU)[6]. The ARPU for big American tech companies has increased considerably over the last four years [Lou13; Lim14; Gar15; Sch16], with Google being an outlier and other companies ranging between \$3.50 and \$20. We chose a conservative estimate of \$7.00 here. The revenue generated per user heavily depends on the business model of the provider and, for providers that are not acting worldwide, on the local users basis, hence and in-depth analysis would need to analyze each provider separately. Data from email marketeers suggests that the revenue third parties generate from email sent to Gmail differs only slightly from email sent Yahoo, Hotmail or AOL [Lan13], which indicates that the wealth or willingness to spend money online is not too different from provider to provider. Nevertheless, the ARPU is likely to vary from country to country, however, we assume the ARPU of the big

[5] While email users are typically not directly profitable for companies, the service is a way to bind clients to the company.

[6] Average margin per user might seem more appropriate, however, it is difficult to obtain figures on the profit of Silicon Valley companies. Moreover, it is very costly to adopt a decrease in users, hence we assume the cost for infrastructure remains unchanged.

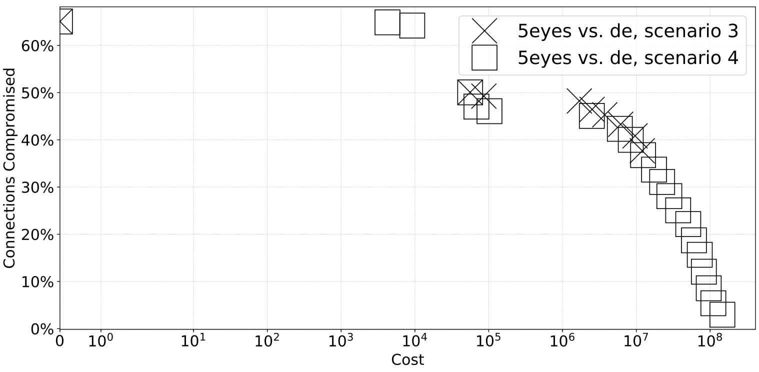


Figure 6.5: Five Eyes vs. Germany (S3 and S4).

Silicon Valley companies is representative for all email providers we investigate, to have a uniform model.

We make the pessimistic estimate that a provider would lose 73% of their users when enforcing TLS (SMTPS or STARTTLS) as opposed to the current opportunistic model. About half of the users targeted in email marketing are using web-mail [Lew17]; we presume only those to produce revenue, as they receive advertisements and are likely to be active on cloud services. Hence we assume $c_{pess} = 0.73 \cdot \$7.50 \cdot 0.5 \approx \$2.56$ per user are lost if TLS is enforced.

This number corresponds to the following scenario. According to a long-running study by Radicati group, Inc., the number of business emails sent and received per user per day totals to 122 emails per day in 2015, among which there 12 spam emails [The15]. These numbers are slowly but steadily increasing despite increased use of IM, chat, social networking and other forms of communication [The17]. In Germany, the number of emails sent is 20 per day [fKon14], which appears consistent. As of May 2017, Google reports that 88% of outbound and 86% inbound emails are already passing standard certificate validation [Inc17], Facebook reported even 95% percent of notification emails in 2014 [Adk14a], Yahoo reported 80% [Ram16]. Taking these numbers as a basis for an average user, 73% is the probability that this average user gives up, if he or she insists that $t_{in} = 90\%$ of inbound emails addressed to them and $t_{out} = 70\%$ of their outbound emails reach their destination.[7] If all users were to behave like this average user, about 73% of users would leave a provider.

In this scenario, TLS is altogether avoided as a mitigation (see Figure 6.5). Throughout the Pareto frontier, only mitigation by relocation appears. The most expensive but effective mitigation is incurring \$15M in cost and requires relocating four name servers of AOL and the email servers of AOL, Microsoft, Google, 1&1 and Arcor, favoring relocation over the deployment of TLS. This is consistent with the observation that, in the wild, TLS connections are not enforced by email providers, which is likely because early adopters would be punished by clients leaving for less secure but more functional services. It is remarkable that DANE/SMTP STS *never* appear in the Pareto frontier

[7] We can compute the probability that this average user gives up using two binomial distributions: $p_{gives\ up} := 1 - \Pr[I > t_{in} \cdot n_{in}] \cdot \Pr[O > t_{out} \cdot n_{out}]$, where $n_{in} = 100$, $n_{out} = 20$ and $I \sim B(n_{in}, 0.86)$, $O \sim B(n_{out}, 0.88)$.

– as in all other scenarios. Upon closer inspection, this is due to the fact that there is only a single route between two MXes (out of 176 relevant routes) that traverses a compromised AS, but where DANE/SMTP STS can be of use because the destination MX is not compromised from the start, e.g., domestic. This route connects the MXes of gmx.net, web.de and 1und1.de (who all belong to United Internet) with t-online. DANE requires setting up a DNSSEC infrastructure, but for t-online, this infrastructure is of little other use (in our model at least), as t-online does not rely on authoritative name servers controlled by the attacker. Hence it is cheaper to set up a VPN for this specific route. Naturally, our routing data is incomplete, but it appears DANE/SMTP STS is of little use when domestic providers are communicating via domestic infrastructure. We disabled RFC 7817, since estimating the cost of its deployment is pure speculation. (We will indulge in speculation in the next section, though.) Consequently, even with an unlimited budget, it is not possible to lower the attacker reward to 0 and 19M transmissions (of $4.95 \cdot 10^{15}$) cannot be considered confidential.

6.7.2.4 S4: deployment cost, optimistic and pessimistic

We take a more optimistic view on the deployment obstacles of TLS enforcement by considering an average user who is perfectly happy as long as half of their incoming emails are received, and half of their outgoing emails reach their recipient. If every provider's user base consisted of only this user, enforcing TLS would cost a provider 3.6‰ of its user base. For RFC 7817, we assume the pessimistic scenario, as it is relatively new (March 2016).

Under these circumstances, enforcing TLS is very useful even in low-cost scenarios (see Figure 6.5). Up to a total cost of $25k, the dominant solution is to enforce TLS with the largest subset (by market share) of German email providers within the budget. Above that, a mix between enforcing TLS and relocating email servers and name servers of US email providers is the most cost-efficient. Enforcing RFC 7817 is only gaining interest starting at a $12M budget. Relocating the email servers of Microsoft, AOL and Google (but not 1&1 and Arcor, as in S3), the adversarial success can be reduced to 0 if all remaining providers enforce TLS with validation according to RFC 7817, however, at the cost of approx. $156M. This confirms that RFC 7817 validation is an effective countermeasure, even if it comes at high cost.

6.7.2.5 S5+S6: the security advantage of deploying RFC 7817

The results in scenario S3 and S4 suggest the following question: If one would go through the pain of enforcing TLS for SMTP traffic, would it not make sense to deploy RFC 7817 right away? We compare those two scenarios by disabling one or the other and comparing the results. RFC 7817 is strictly stronger but requires some additional effort at the recipients' side. It is hard to estimate this cost, but our comparison gives an indication of the cost saved by other measures becoming redundant due to RFC 7817's stronger resistance against attacks on the DNS level.

In the optimistic scenario (see Figure 6.6), our data suggest that not only RFC 7817 is the better mitigation from the start, but starting from a budget of approx. $4 000, the attacker reward is consistently 20 percentage points higher if TLS is enforced (w/o

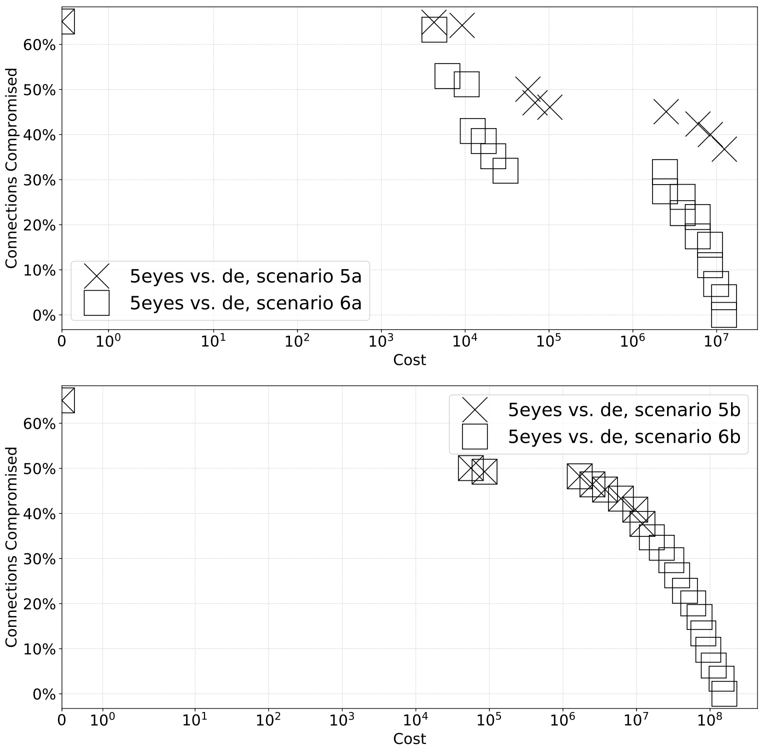


Figure 6.6: Five Eyes vs. Germany (S5a and S6a) & Five Eyes vs. Germany (S5b and S6b).

RFC 7817) for the same budget. Again it is not possible to lower the attacker reward below 19M connections, whereas validation according to RFC 7817 can lower it to zero.

In the pessimistic scenario (see Figure 6.6), however, there is almost no difference between both Pareto frontiers, because in both scenarios, relocation of NS servers and IPsec is preferred to enforcing TLS, be it with or without RFC 7817 certificate validation: TLS is never enforced in the Pareto frontier to S5b. TLS+RFC 7817 appears in S6b at the cost of ca. \$14M, however, it is only slightly more effective than the corresponding solution by relocation in S5b.

6.7.2.6 S7: hidden attacker, crouching DNSSEC

DNSSEC can also be used as a forensic tool, as signatures permit identifying misbehaving parties and provide a time stamp to indicate responsible actors [SW14]. As our motivation is an attacker who attempts large-scale surveillance without getting caught too often, it is interesting to look at the scenario where the integrity of a domain is protected by DNSSEC even if a name server used during resolution (and thus part of the trust chain) is controlled by the adversary, because he or she is trying to avoid exposure. This amounts to simply adding $\mathsf{nDNSSEC}(n)$ to the premise of rule r_{dns-ns} (see Section 6.5.1).

Comparing the Pareto frontier in S4 and S7 (see Figure 6.7), we observe no difference from the start, but a significant increase of confidential connections in the mid-cost region. The attacker reward is lowered to zero one order of magnitude earlier. With increasing

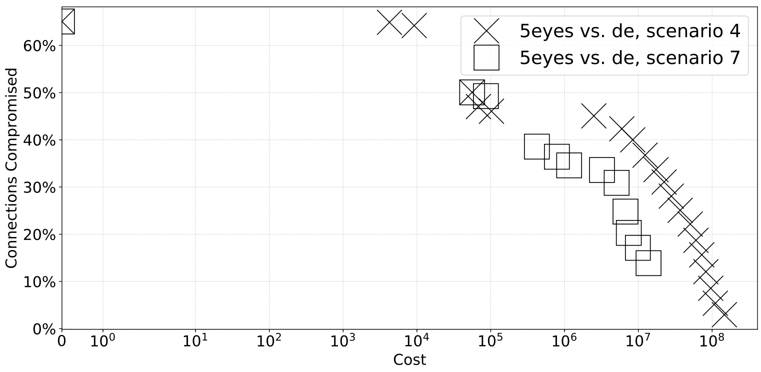


Figure 6.7: Five Eyes vs. Germany (S4 and S7).

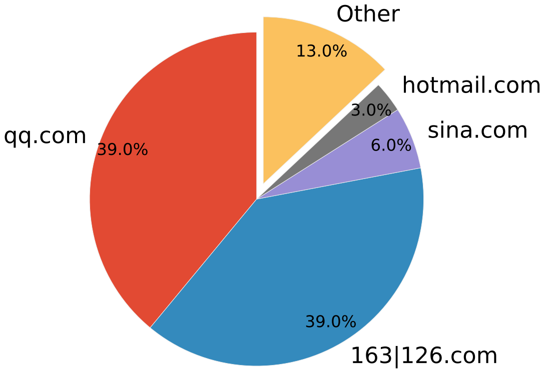


Figure 6.8: Most used (primary) email providers in China [Zij11].

budget, first the IPsec connection for the aforementioned route (see S3) is used, then domestic providers deploy DNSSEC, except for t-online (which has little dependency on compromised name servers) and freenet.de (where it is cheaper to relocate the few name servers they have). As DNSSEC now also mitigates DNS spoofing via authoritative name servers, it becomes a cheaper alternative to strict TLS enforcement at all budgets. This means that DNSSEC is a very cost-efficient and effective countermeasure against large-scale email sniffing, provided that the attacker can be held responsible for misbehavior or has other reasons to avoid exposure that cannot be denied easily. Again, DNSSEC is not deployed, as it is cheaper for t-online to secure the single malicious route via IPsec than to use DANE/SMTP STS and pay the additional cost of deploying DNSSEC.

6.7.3 Results for Other Countries

We now turn our attention to the other defender countries.

Figure 6.8 - Figure 6.11 list the most popular email providers by country based on the market share they possess.

For space reasons, we limit the discussion to the fourth cost scenario (S4), which allows to compare the effect of all defender actions. Figures 6.12 to 6.16 give a combined view of the respective defender countries against the different attacker countries. Attacker

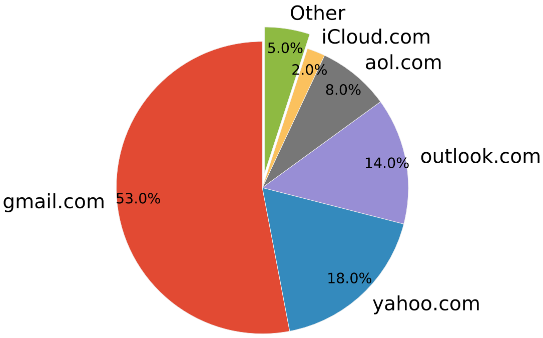


Figure 6.9: Most used (primary) email providers in USA [Ade16].

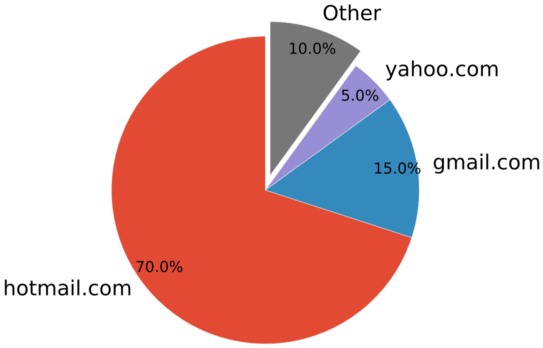


Figure 6.10: Most used (primary) email providers in Brazil [Vah13].

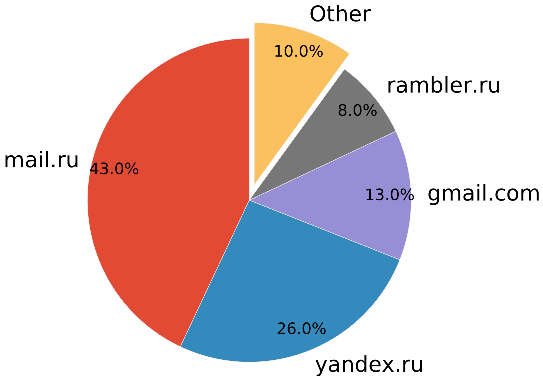


Figure 6.11: Most used (primary) email providers in Russia [Kon17].

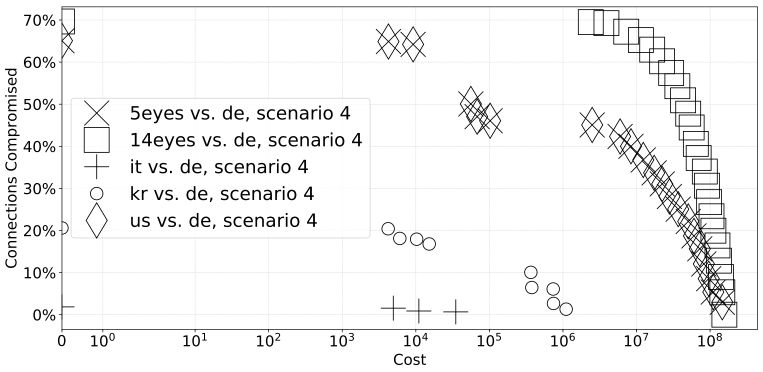


Figure 6.12: Results for defender country Germany (S4).

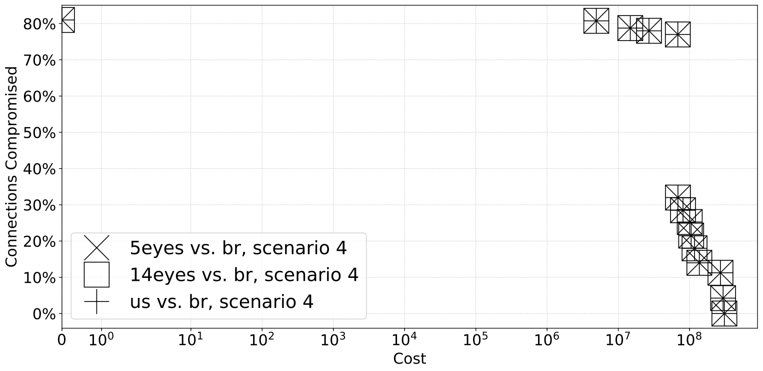


Figure 6.13: Results for defender country Brazil (S4).

countries which cannot compromise any email communication of the considered defender country have been left out, and coalitions are excluding the defending country.

6.7.3.1 Germany

Among the Five Eyes countries, only the US can affect email confidentiality of German users. In low-cost scenarios below \$56k, enforcing TLS provides a remedy, above this budget, and up to \$151M, securing the connection between two ASes via IPsec is an effective measure. To completely defend, it is necessary to relocate the email servers of the American providers, and to enforce TLS with RFC 7817 for all providers to protect from TLD servers hosted in the USA. Extending the attacker to all Fourteen Eyes countries leads to additional attacks on domain resolution, significantly increasing the necessary mitigation budget. Relocating all compromised email servers, and enforcing TLS with RFC 7817 certificate validation, however, is still sufficient to protect Germany's email confidentiality. Finally, South Korea and Italy are able to compromise email confidentiality through malicious routes, but only to marginal success. For South Korea, only DNSSEC and TLS enforcement appear in the Pareto frontier, for Italy only relocation and TLS enforcement.

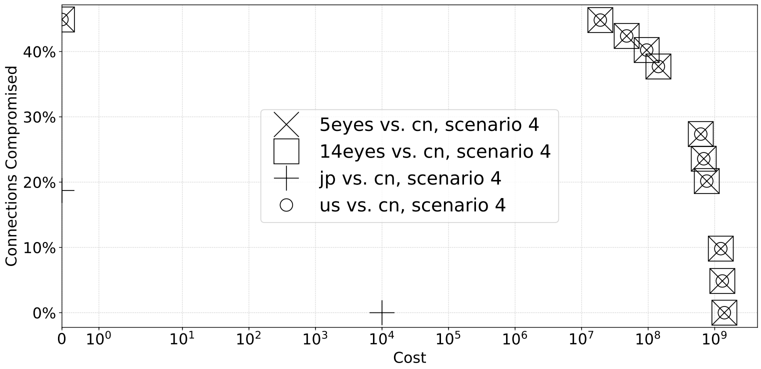


Figure 6.14: Results for defender country China (S4).

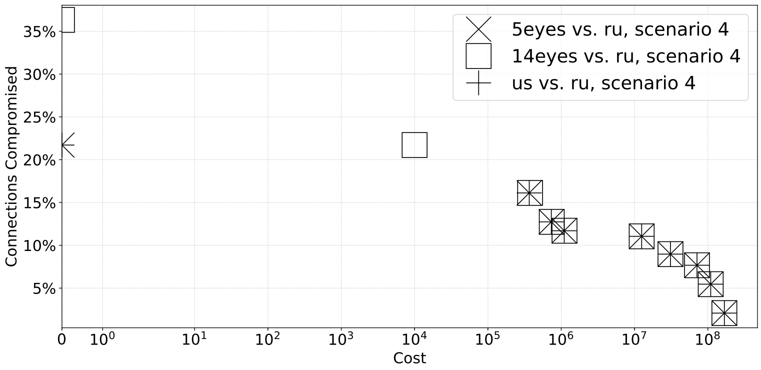


Figure 6.15: Results for defender country Russia (S4).

6.7.3.2 Brazil

According to our statistics, US email providers entirely dominate the email market of Brazil [Vah13]. Consequently, Brazil has a significant dependency on US infrastructure. Relocating email servers of the less used providers (Yahoo and Gmail) along with enforcing TLS with RFC 7817 is the cheapest mitigation strategy. Starting at 68M, relocating the email servers of Hotmail, being by far the most prominent provider in Brazil, leads to a tremendous decrease in compromised connections.

6.7.3.3 China

Only the US and Japan are relevant attacking countries in this scenario. To mitigate attacks from Japan, it is enough to relocate a single name server. On the other hand, to mitigate attacks from the US, the best solution is to relocate the email servers of Hotmail and to enforce TLS and RFC 7817 on the domestic providers. Note that DNSSEC seems irrelevant because we lack information about routing, and mostly Chinese TLDs are used.

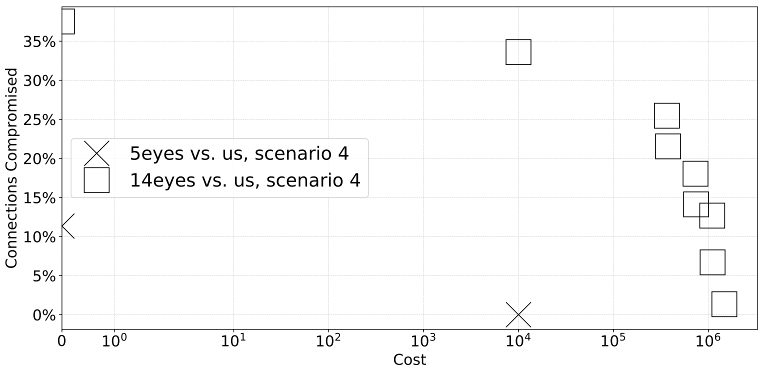


Figure 6.16: Results for defender country USA (S4).

6.7.3.4 Russia

Apart from gmail.com, Russian users rely on domestic email providers. This is the reason why there is no country or alliance that can compromise more than ca. 36%. Relevant attackers are the US and Fourteen Eyes. The only difference between them is that against Fourteen Eyes a single compromised name server needs to be relocated. The immediate and the cheapest mitigation is to activate DNSSEC on the three domestic providers because they use US controlled TLDs. Further, relocating the email servers of Gmail and activating DNSSEC on only two domestic providers is a close second. To reach confidentiality for all communication, it is necessary to enable DNSSEC and enforce TLS with RFC 7817 on all domestic providers and to relocate Gmail's email servers.

6.7.3.5 USA

To defend from other countries which are part of Five Eyes, it is necessary for the US to relocate a single name server which resides in the UK. For defending from the Fourteen Eyes countries, more action is needed. Strictly speaking, all five widely used domestic email providers need to enable DNSSEC because some routes are traversing foreign countries.

6.8 Related Work

6.8.1 Infrastructure Analysis on the Internet

Many works have been studying global and targeted attacks on the Internet infrastructure and the reasons behind these attacks [Gol14; BFM$^+$10; SPR$^+$17b] but the systematic evaluation of mitigations has largely been neglected. Frey et al. [FER$^+$16] studied the European BGP topology w.r.t. to disruption scenarios, but considered only three possible outcomes from mitigation. Simeonovski et. al. [SPR$^+$17b] presented a model of the Internet infrastructure using property graphs to assess attacker impact, but not possible mitigations. We extend their property graph for modeling the Internet infrastructure and adopt a completely new and more expressive attacker model (adding, e.g., name resolution and routing).

6.8.2 Symbolic Soundness of Threat Model

Dax and Künnemann [DK21a] proposed a methodology to justify a rule-based attacker model by adding another layer of abstraction on top of the symbolic model of cryptography, which reasons about protocols and abstracts cryptographic primitives. They show, in general, how the soundness and completeness of a model can be ensured by verifying trace properties, linking soundness to safety properties and completeness to liveness properties. They then demonstrated their approach for the threat model presented here (in Section 6.5.1). They showed that it provides symbolic soundness after fixing two minor flaws in the model. The first flaw is the missing rule $r_{init-ip}$ which is already corrected in dissertation. The second flaw is DNSSEC as a mitigation against rule $r_{dns-route-res}$ which we decided to keep to investigate the potential effectiveness of DNSSEC in the future.

6.9 Conclusion

We showed that a holistic analysis of deployment benefits of cryptographic protocols, secure configurations and political measures is possible with a high degree of automation. Based on a very simple cost assessment in the case of email communication, we see that the enforcement of TLS would have a great effect in most countries. If strict TLS validation can be achieved, RFC 7817 should be implemented with it: despite its simplicity, it reduces mitigation cost by more than 20%. However, it is plausible that the hidden cost of deployment, a loss of functionality, make this approach impractical. Techniques like DANE and SMTP STS do not suffer this problem, however, in the case of Five eyes vs. Germany, they are never of use. This might be different for Russia vs. US and US vs. Five eyes, where DNSSEC is employed a lot, but the lack of routing data prohibits a conclusion in these cases. DNSSEC itself is particularly useful in the scenario where the attacker does not tamper with the trust chain. There it substitutes enforcement of TLS completely. Such *sneaky* adversaries should receive further analysis in the future. Even if the adversary is not sneaky, DNSSEC is useful when the domestic infrastructure is relatively self-sufficient, e.g., for Russia and the US in scenario S4. While the US, Russia and China are relatively self-sufficient, the privacy of email users in Brazil is highly vulnerable due to foreign dependencies, requiring costly relocation.

An interesting avenue for future work is to capture probabilistic threat models, e.g., off-path attacks on DNS that succeed only with a certain chance, vulnerability assessment of old operating systems, or the chance of a user becoming victim to fishing attacks. Probabilistic planning provides a fitting framework for this endeavor and could help, e.g., to prepare emergency responses to large-scale incidents like the recent spreading of the WannaCry worm. Furthermore, the rules defining our threat model are ad hoc. Similar to how abstractions of cryptographic primitives in the Dolev-Yao model are justified via computational soundness, it is worth exploring how holistic threat models for risk assessment, which have to be somewhat sound and complete, can be derived from protocol specifications. Most importantly: precise cost estimates are hard to come by, in particular the cost induced by loss of functionality and interoperability. We hope that the methodology presented here encourages the in-depth analysis and validation of deployment cost, both for existing and for future security mechanisms. In case an interested reader wants to play around with their own cost estimations and wants to investigate the results of this case study: we present a web-based GUI for that in Part IV.

The email infrastructure is not the only successful Internet-wide application for Stackelberg planning. We tackle security protocols for the web in the following chapter.

7 Formally Reasoning About Pareto-Optimal Defensive Strategies for Securing the Web

7.1 Motivation and Contributions

Billions of people make use of the web on a daily basis for business and private life. Given this success of the web as a platform, the impact of attacks on the web is enormous. Users can be unconsciously forced to visit a phishing version of their bank website, redirected to an exploit kit by means of drive-by download attacks, execute scripts to mine cryptocurrency or perform DDoS attacks. Securing the user's activity on the Web is a serious challenge: not only do servers hosting a domain's content need to be protected from compromise, but the reliance of many sites on external JavaScript means that a compromised third-party will affect the including site's security. Moreover, the Internet's infrastructure plays a key role in securing a domain. This infrastructure covers resolution of domains to IP addresses and routing of IP packets between different hosts. On top of this, TLS can be used to ensure confidentiality and integrity of the transmitted data, which in turn relies on an uncompromised Public Key Infrastructure (PKI) system.

Securing a website, therefore, is not something a site operator can achieve on their own. Instead, actors like Internet service providers, Internet exchanges, name service providers, content delivery networks, and certificate authorities influence the whole ecosystem. Thus, the security of the web ecosystem hinges on the infrastructure and all involved actors as a whole.

In this chapter, we present a methodology to evaluate existing security proposals against mass attacks on the Web.

Various proposals have been made to improve the security of the Internet infrastructure in terms of routing (IPsec [KS05]), name resolution level (DNSSEC [AAL+05a]), website delivery (HTTPS [Res00], HSTS [HJB12]), public-key infrastructure (certificate transparency [LLK13], DANE [HS12]), and third-party JS inclusions (CSP [BS15], SRI [WBA+16]). They all raise the level of security, but which combination of proposals is the most cost-effective, *considering the current infrastructure*? Are some of them too costly to deploy or simply less effective than existing proposals? We proposed *Stackelberg planning* in Part I to answer these questions, a two-level planning problem where a defender is given a budget and a choice of *mitigation actions* to find the most effective combination of these actions to lower the maximum success of an attacker who themselves is combining attack actions to compromise specific targets. We successfully applied Stackelberg planning to the security of the email infrastructure in Chapter 6 where the application layer is simple in comparison and a large part of the population is typically served by a handful of email providers. In contrast, web security needs to consider users accessing thousands of domains, thus presenting a problem at a completely different scale.

To close this research gap, we developed an alternative approach to solving large-scale Stackelberg problems, based on the graph database system Neo4J[1]. Our methodology exploits three features of this particular problem: (1) The planning task is relaxed, meaning that no action the attacker performs can block another action and thus backtracking is not necessary. (2) The dependencies between actions are largely acyclic, meaning that we can minimize the need for fixpoint computations and use Neo4j to perform all actions of a certain class (e.g., machine-in-the-middle attacks on JS inclusions) in bulk. (3) We want to compare a relatively small number of mitigation strategies applied to a relative large system; hence the ability to reuse information between different mitigation scenarios can be traded off for a more efficient computation of the attacker's success in a given scenario. With this methodology, we can scale up to about 71k infrastructure elements and 2.2M attacker actions.

Our second contribution is a comprehensive threat model for web-based attacks, which covers both aspects of the underlying infrastructure and web attacks themselves. Based on this threat model, we build a defender model which considers different defensive actions, their associated costs, and potential dependencies for deployment (e.g., DANE requires DNSSEC to be secure). With these models, we can compare competing proposals (e.g., the use of SRI versus secure HTTPS inclusions) w.r.t. the infrastructure.

Our third contribution is a comprehensive analysis of these mitigation strategies securing clients that visit the Alexa Top 5k. We consider three classes of attackers in terms of their arsenal and capabilities: cyber-criminal groups [tea18], malicious infrastructure providers for cloud services and name resolutions, and three state-sponsored attackers. For each, we compute the cost and the efficacy (in terms of reducing the number of affected visits on the Alexa Top 5k) of the existing deployments of that technology, and the cost and efficacy of the potential deployments yet to come. We created a web-based GUI at mitigation-web.github.io to analyze and investigate optimal mitigation deployments with customizable costs.

[1] https://neo4j.com/

Goals of this chapter: We provide a methodology to evaluate the cost-effective selection of mitigations for the entire Internet infrastructure as the result of global policy that aims at making the web secure for all users. We analyze the effect of mitigations that can avoid or limit the attacker's ability to affect user visits on websites.

Not in scope of this chapter: We do not focus on the greater goal of the attacker neither on ad hoc defenses for individual hosts or companies. The results of exploiting weaknesses of the Internet infrastructure can range from cryptojacking to phishing, attacks against password managers, or DDoS [KVM$^+$18; HYY$^+$18; SJ14; SJB$^+$14; OR20; MWD$^+$15] depending on the attacker's motivation and fall outside the scope of this chapter.

7.2 Stackelberg Planning Recap

As a recap, we proposed *Stackelberg planning* in Part I which can be seen as a two-fold classical planning task. Inspired by work on Stackelberg security games, the defender moves first and the attacker is able to fully observer the defender's actions and can plan their best response accordingly. In our notion of Stackelberg planning here, the actions of the attacker have an *attacker reward* which is used as an indicator of the severity of the attack. Instead of a *plan* leading to a *goal state*, the set of *attacker actions* maximizing the attacker reward is computed, e.g.the number of compromised domains. To prevent attacks and thus to lower the attacker reward, the defender can change the world state through the application of *defender actions*, also referred to as "mitigations" which are assigned a cost. The defender pursues the objective to simultaneously minimizes its own cost and the *attacker reward for the resulting state after applying the defender plan.*

If the attacker can, e.g.get hold of 10 domains, the defender weighs the damage done against it own cost. We avoid fixing this weight by instead considering the *Pareto frontier*, i.e.the set of all Pareto optimal defender plans. A plan is dominated by another plan, if the second plan is either cheaper (strictly lower cost) but as effective (lower or same attacker reward), or vice versa (lower or same cost, strictly lower reward). Any plan that is dominated by no other plan is Pareto optimal and thus part of the Pareto frontier. Thus the Pareto frontier gives us the set of all defender plans that are economically reasonable, i.e.optimal for their respective budgets. It also gives us a step function from budgets to the level of security, i.e.the remaining attacker reward, that is achieved by the optimal plan for this budget. Our formal model introduced in Section 7.3 and Section 7.4 is a Stackelberg planning task, but instead of using algorithms from Part I to compute the solution (i.e., the Pareto frontier), we introduce a novel graph-based algorithm in Section 7.7.

7.3 Threat Model

We focus on infrastructure attacks here, i.e.those that arise from physical, logical and administrative dependencies in the Internet, as opposed to weaknesses in protocol

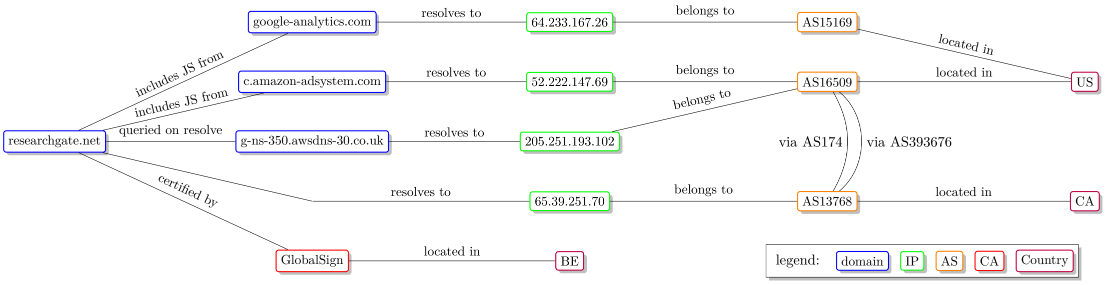


Figure 7.1: Infrastructure relations.

specification or in the implementation.We therefore assume that protocols and web mitigations achieve their stated goals, e.g., provide a secure communication channel, but the attacker may break the trust assumptions, e.g., when a Certificate Authority (CA) is compromised.

Our threat model consists of a set of *attacker rules* which are listed in paraphrased form in Table 7.1 and formally defined in Section 7.5. These rules describe a layered model in which we depicted the different attacks that can be carried out for each layer: routing level attacks can be used to compromise the integrity of packet transmission, DNS-level attacks can compromise the integrity of the name resolution and application-level attacks can compromise the content on the website. The combinations of these attack vectors can be used to compromise the visits on a specific website.

7.3.1 Infrastructure

To illustrate the most important infrastructure dependencies in the web, consider the following common place example (Figure 7.1). A user browses through a gallery on the popular image hosting service researchgate.net. The browser first has to resolve this domain to an IP. This is done through a series of DNS queries performed by the resolver (usually first the user's local resolver, then its ISP's) to contact the authoritative NS. The correct resolution of researchgate.net depends on each of those.

After resolution, the user connects to an IP, which belongs to an autonomous system (AS). These ASes are interconnected, and packets need to be routed via multiple ASes. On the network layer, the integrity of the packet transmission depends on each AS that is traversed. Each AS is associated with a country, which we use to model attacks by state actors.

The website can now be delivered but it might include JS from external websites, in this case google-analytics.com, which in turn depends on various name servers, on the AS the IP belongs to, etc.

In case the website is retrieved via HTTPS, the authenticity of the connection depends on the signing CA and *all* root CAs whose certificates come with the user's browser, as any of these may supply the website's certificate.

Table 7.1: Paraphrased attacker actions associated to class of attackers (nation-state (N), service providers (S), small hacking groups (H)), paraphrased (formal definition in Section 7.5).

	#	attack vector	precondition	outcome	applicable mitigations	class of attackers
initial compromise	(1)	attacker control	country compromised and entity (AS,IP,name server or CA) associated to this country	entity compromised	none	N
	(2)	attacker control	AS compromised and IP i belongs to AS	i compromised	none	N,S
	(3)	attacker control	IP i compromised and domain d resolves to i	d compromised	none	N,S,H
	(4)	attacker control	domain d compromised and d resolves to IP i	i compromised	none	N,S,H
routing	(7)	routing compromise	AS$_2$ potentially en route from AS$_1$ to AS$_3$ and AS$_2$ compromised	routing from AS$_1$ to AS$_3$ compromised	IPsec	N,S
	(8)	routing control	AS$_1$ compromised	routing from AS$_1$ to AS$_2$ compromised	none	N,S
DNS	(6)	DNS poisoning	name server d' queried when resolving d and d' compromised	resolution of d compromised	none	N,S,H
	(10)	DNS hijacking	name server d' queried when resolving d and d' in AS$_2$ and AS$_1$ geolocated in country and routing from AS$_1$ to AS$_2$ compromised	resolution of d from country compromised	DNSSEC on d'	N,S
certificate compr.	(16)	certificate spoofing	some CA is compromised	certificate of d can be forged	Certificate Transparency* (on d's CA); DANE (on d's authoritative NS)	N,S
	(17)	DANE record spoofing	some CA is compromised and d' authoritative for d and d' compromised	certificate of d can be forged	Certificate Transparency* (on d's CA)	N,S
	(18)	trust chain compromise	CA a is compromised and TLSA assumes trust in a	certificate of d can be forged	Certificate Transparency* (on d's CA)	N,S
content	(5)	XSS	XSS vulnerability on d	website on d compromised	none	N,S,H
	(9)	website MITM	Domain d resolves to IP in AS$_2$ and AS$_1$ geolocated in country and routing from AS$_1$ to AS$_2$ compromised	access to website on d from country compromised	HTTPS + HSTS + HTTPS-Redirect (unless certificate of d can be forged)	N,S
	(11)	from DNS poisoning	resolution of d compromised	website on d compromised	HTTPS + HSTS + HTTPS-Redirect (unless certificate of d can be forged)	N,S,H
	(12)	from DNS hijacking	resolution of d from country compromised	access to website on d from country compromised	HTTPS + HSTS + HTTPS-Redirect (unless certificate of d can be forged)	N,S
via CDNs/JS inclusion	(13)	from DNS poisoning	resolution of d' compromised and d includes JS from of d'	website on d compromised	SRI (res. from d'); secure incl. (res. from d') (unless cert. of d' can be forged); HTTPS + HTTPS-Redirect (unless cert. of d' can be forged); upgrade-insecure-requests on d (unless cert. of d' can be forged)	N,S,H
	(14)	from DNS hijacking	resolution of d' from country compromised and d includes JS from of d'	access to website on d from country compromised	SRI (res. from d'); secure incl. (res. from d') (unless cert. of d' can be forged); HTTPS + HTTPS-Redirect (unless cert. of d' can be forged); upgrade-insecure-requests on d (unless cert. of d' can be forged)	N,S
	(15)	via routing	d includes JS from d' and AS$_1$ located in country and d' within AS$_2$ and routing from AS$_1$ to AS$_2$ compromised	access to website on d from country compromised	SRI (res. from d'); secure incl. (res. from d') (unless cert. of d' can be forged); HTTPS + HTTPS-Redirect (unless cert. of d' can be forged); upgrade-insecure-requests on d (unless cert. of d' can be forged)	N,S
	(19)	3rd-party JS-incl.	d includes from d' and d' is compromised	website on d compromised	SRI for resources from d'	N,S,H
	(20)	website compromised	d is compromised	website on d compromised	none	N,S,H

* Mitigation due to sneakiness assumption.

7.3.2 Class of Attackers

We considered three classes of attackers with different capabilities: small cyber-criminal groups, malicious service providers, and nation-states. Each class has access to a given set of compromised entities, e.g.ASes, IPs, websites, CAs, or NSs that translate into a subset of rules described in Table 7.1. Not all of the attack vectors are available to all classes of attackers as some are traits specific to particular attackers. In Table 7.1 we identified which classes hold the capability for each attack vector. This will be used in the analysis in Section 7.8. We underline that this assignment is not definitive as, e.g., small hacker groups can also potentially compromise CAs, but our framework allows to define different scenarios of adversaries targeting users of the web.

We evaluate the impact of an attacker in terms of the number of websites it can compromise, weighted by the number of users visiting these websites, i.e., the attacker plan maximizing $\sum_{i \in countries} Visitors_{i,d}$ for $Visitors_{i,d}$ being the estimated number of visitors for the Website d from the Country i.[2]

By computing the maximum attacker reward, we can measure the potential impact of web attacks on the Internet and the efficacy of the mitigations in scenarios characterized by the initial assets of the attacker, the set of rules available to the attacker, and the countries defended.

For the class of attackers considered, the stealthiness is of the utmost importance to avoid attribution and retaliation [DOJ21], in particular for service providers and countries. Therefore, for a first approximation, we ignored attacks that can be easily detected and that can result in a global exposure to a company or country, e.g.BGP hijacking attacks. Hence, our attacker is ‘sneaky’.

7.3.3 Attacker Rules

The threat model is described in terms of attacker actions that are instances of the rules in Table 7.1. The state predicates capture which entities (ASes, IPs, domains, CAs, NSs) exist, how they are related and which mitigations have been deployed, but also the state of the attack. The state of the attack is represented by the following predicates:

- An entity can be compromised globally, or for users from a country, in which case it can deliver malicious content.
- A route between two ASes can be compromised, in which case the attacker can inject/reroute packets on this route.
- The DNS resolution of a domain can be compromised (for all users or for users from a country), in which case the attacker can manipulate DNS queries for this domain.

The complete model and the list of predicates are presented in Section 7.5 along with the intuition for each rule. Here, we only consider an example for illustration. Say we consider China as an attacker mounting a Great Cannon-like attack, i.e.Chinese authorities intercept requests to included JavaScript resources and modify their content [MWD$^+$15]. Suppose users visit the popular website www.diply.com. By rule (1),

[2] We use the number of visitors per month as retrieved from Alexa, thus we are assuming the attacks to be stealthy and to persist for some time. Furthermore, we ignore countries that constitute less than 0.01% of the website's visitors.

China controls the AS714[3], over which packets from Japanese users may be routed when contacting a10-67.akam.net, which is in AS21342. By rule (7), this route is compromised. a10-67.akam.net is the authoritative NS for cdn.diply.com, the resolution of this domain is considered compromised by rule (10). As www.diply.com includes JavaScript code from cdn.diply.com, this website is vulnerable to JavaScript injection via DNS spoofing (rule (14)).

7.3.4 Discussion

As our analysis measures the efficacy of mitigations in terms of adversarial success, it needs to be as precise as possible, ideally capturing all attacks, and only those attacks. Dax and Künnemann outline how to establish soundness and completeness w.r.t. a Dolev-Yao attacker interacting with the protocol *according to specification* [DK21b], but we consider this out of the scope and take this attacker model as granted. Like the Dolev-Yao model, our model assumes the absence of implementation-specific errors induced by the user, which could at best be estimated at this point.

We weigh the domains by the number of visits to reflect their popularity. This is not a measure for the number of users that can potentially be infected, as the reward is additive and thus counts visitors that frequent two domains twice. Some domains likely share more users with each other (*thehackernews.com* and *wired.com*) than others (*google.com* and *bing.com*). To compute the number of infected users, we would need information about the intersection of visitors, ideally for all sets of Alexa-listed domains.

If the attacker has access to one of the NS potentially queried in the name resolution of a domain, the integrity of the resolution is considered compromised. As there can be more than one authoritative NS per domain and caching may prevent the iterative resolution, this is an over-approximation. Similar, we consider a route between two ASes compromised if either of the endpoints is compromised, or if a compromised AS is potentially en route. Additional inaccuracy is introduced by the fact that routes change over time, see Section 7.6.1 for how potential routes are acquired.

We assumed the attacker wants to avoid global exposure due to the forensic evidence. As a result, we consider attacks against the PKI as mitigated if the target domain's certificate was signed by a CA compliant with Certificate Transparency. We showed that this assumption does not impact the results by analyzing the case in which CT is disabled and DANE is applied instead. Similarly, we exclude BGP hijacking and attacks on the DNS root servers. For BGP hijacking, similar results can be obtained by considering attacks at the network layer. Furthermore, it would require assumptions on how the (sub-)prefix hijacking and the BGP routes propagate. We leave the implementation of these attacks for future works.

7.4 Mitigations

The defender model consists of a set of actions that aim to minimize the attacker reward by implementing a set of mitigations. Each defender action has an associated cost and mitigates one or more attack vectors. We gather the cost based on publicly available

[3] Some prefixes are partially used at this location.

data and try to keep the cost model uniform, i.e.we do not take differences in cost of labor due to the location or structure of the company into account. E.g., youtube.com and google.com belong to the same company, but only recently announced they will share infrastructure [Afi21]. We detail our cost model in Section 7.4.6, but stress that a uniform cost model, while being easy to convey, can never exactly represent the actual operating cost in such a diverse set of companies as the Alexa Top 5k. Moreover, what to include as direct cost of a technology like DNSSEC is very much debatable. We therefore provide a website in which the cost can be tuned for the specific needs at mitigation-web.github.io.

We now present the mitigations that can be applied at different levels of the Internet infrastructure.

7.4.1 Application Layer Mitigations

7.4.1.1 SRI

CDNs are a valuable target for attackers, as thousands of websites often depend on a particular resource they host, e.g.widely used libraries like jQuery. A modification of this resource can infect the users of the including website. With Sub Resource Integrity (SRI), the including website provides the hash value of each resource hosted on a third-party server with the script tag. The browser then compares this hash value to the hash value of the retrieved file. If the values do not match, the browser does not execute the resource.

This type of attack is widespread and can be implemented in large scale as shown by the *Great Cannon* attack [MWD+15; Ano14]. The implementation of SRI for the resources retrieved from Baidu could have reduced the impact of this attack [Toe15].

Although the adoption of SRI is growing [COC+20], it is not suited for resources that change over time, e.g.versioned JS libraries, or dynamically generated scripts.[4] This scenario is not uncommon, however, it is often caused by minor changes (e.g., recompilation) that can be easily avoided [SMJ+21].

7.4.1.2 Other Mitigations

Another mitigation could be Content Security Policies (CSP) [SSM10]. For a first approximation, we decided to not consider CSP for mitigating XSS in our model because the adoption is currently strongly limited by the required cooperation with third-parties [SMJ+21], with the result that most of the deployments are insecure and enable inline scripts [WSL+16; CRB16; WLR14]. Given that CSP is mainly used to prevent inline XSS [eal21; WLR14] and does not prevent other attack vectors available for our classes of attackers (e.g., compromise of whitelisted CDNs), we are confident that CSP would not affect the overall results. We discuss the extension of the model in Section 7.10.

[4] To temporarily handle this mismatch, the external resource can be retrieved from a local repository.

7.4.2 Transport Layer Mitigations

7.4.2.1 TLS

The HTTP connection between a client and a website can be secured through TLS to achieve authenticity, integrity, and confidentiality. At the time of writing, HTTP is the default protocol in almost[5] all major browsers. As we assume users to not specifically ask for HTTP over TLS (HTTPS) connections, websites need to implement a redirect and set an appropriate HSTS header (see below) for this mitigation to be effective.

7.4.2.2 Redirects and HSTS

While a secure redirect is not a mitigation in itself, it is necessary to provide a secure connection for the exchange of an HSTS policy through the strict-transport-security header. Indeed the header is ignored in an HTTP connection[HJB12]. On the other hand, to ensure that any further access to the server is directly conducted over HTTPS, it is necessary to implement an HSTS policy.[6] An HSTS policy is an HTTP header that informs the browser that the specific domain and (if explicitly declared) its subdomains must be accessed via HTTPS for a certain period of time. All major browsers come with an *HSTS preload* list that contains a set of domains for which the browser automatically creates an HTTPS connection. However, it is required to keep a HSTS header to maintain the domain in the preload list.[7]

7.4.2.3 Secure Inclusions and CSP upgrade-insecure-requests Directive

To secure inclusions from third-party websites, subresources should be loaded through a secure connection, either explicitly specifying the HTTPS protocol or using a Content Security Policy with an upgrade-insecure-requests directive. The latter informs the browser that all the site's insecure URLs must be replaced with HTTPS.

An attacker can exploit subresources retrieved through HTTP by conducting a MITM attack. This scenario is limited to the case in which the main web page is loaded over HTTP as currently modern browsers block *mixed content* for active resources. We stress that different browsers handle mixed content differently, and outdated browsers might still be vulnerable to this attack. We reserve a closer look at how legacy browsers change the picture for future work and assume all browsers to block active mixed content.

[5] Only the most recent version of Chrome [Blo21a] and Firefox in private mode [Blo21b] use HTTPS by default. Safari defaults to HTTP.

[6] Under rare circumstances, a redirect can increase security by itself: if a network injection attack is possible on an included resource, a redirect ensures that the malleable resource is not loaded because it would constitute mixed content (see Rule (15)).

[7] https://hstspreload.org/#continued-requirements

7.4.3 Routing Layer Mitigations

7.4.3.1 IPsec

To prevent attacks at the network layer from a malicious AS in the path between two ASes, packets routed between the two ASes can be encrypted and authenticated through IPsec with a gateway-to-gateway architecture [KS05; Hou05]. We assume the implementation of an IPsec connection to not be influenced by the geolocation of the endpoint and to be the result of a private agreement between AS owners.

7.4.4 Resolution Mitigations

7.4.4.1 DNSSEC

To prevent DNS spoofing attacks, DNS records can be authenticated with DNSSEC [AAL+05a]. The adoption by end users is still very low [APN], but it can be implemented in the recursive resolver of the ISP [NA10]. We assume this and that the route from the user to the recursive DNS resolver is secure. The latter assumption is necessary, as we do not have data on how the visitors reach their recursive resolver and the opposite assumption would render DNSSEC useless. As we will see in Section 7.8, DNSSEC achieves modest security improvements despite this over approximation.

We consider this mitigation for all domains where all the parent domains up to the root already support DNSSEC. At the time of writing, DNSSEC is deployed in the root servers and in more than 90% of the TLDs [ICA21].

7.4.5 CA Mitigations

7.4.5.1 Certificate Transparency

The authenticity of a web server on the Internet relies on digital certificates issued by certificate authorities. In the last years, this model showed many flaws including mistakenly issued certificates and CA compromise. Google proposed the Certificate Transparency (CT) [LLK13] project as a measure to detect misissued certificates; this is done through a set of publicly available append-only certificate logs that contain all the certificates present on the Internet. CAs must submit the certificate to a log to receive a signed certificate timestamps required by the browser during the TLS handshake. Domain owners can verify the list of digital certificates issued for their domains and detect the presence of unauthorized ones. Chrome requires all certificates issued after 30 April 2018 to be compliant with the CT policy and Safari requires signed certificates timestamps. Given that Chrome and Safari alone cover more than 78% of the desktop browser market share [Sta], and that Mozilla is planning to include support for the CT project [MDN21], we assume the entire CA ecosystem to be CT compliant. Given that our threat model (Section 7.3) considers stealthy attacks, we ignored the scenario in which an attacker issues malicious certificate via a compromised CA. Nevertheless, we investigate the effect of DANE as an alternative to CT in a separate scenario.

Table 7.2: Mitigation cost per host. Let $r = 140\,\$/\mathrm{h}$ the daily rate of an external consultant.

Mitigation	Cost per host	Comment
SRI	r*8h	Consultant cost for 1 day. Existing tools to support (e.g.[Jus16]). We do not consider any backup cost to handle mismatches of hashes. Although SRI requires CORS to be enabled for included resources, the cost for setting up this header is negligible, since it requires a single HTTP header to be set[SMJ+21].
TLS	r*8h	Consultant cost for 1 day to modify the web server configuration to allow HTTPS connections (including the effort of obtaining a certificate). We do not include the cost of the digital certificate, given free CAs like *Let's Encrypt*.
Redirect / HSTS	r*8h	Consultant cost for 1 day.
Secure inclusions / UR	r*8h	Consultant cost for 1 day to check that all subresources are available via HTTPS.
IPsec	$56,000 per link	Cost for a link speed of 10 Gb/s. Including the cost of two dedicated routers for $24,000 each [RGE+16] and the consultant cost for configuration and maintenance per year (about 80 consulting hours) [Cha17].
DNSSEC	$366,342	Cost of deploying in all the authoritative NS managed by a company based on the maximum CAPEX from a survey [NA10], one of which appear in the Alex Top 100.
CT	$0	The CA ecosystem is already CT compliant and mitigation can only be applied if TLS is already deployed.
DANE	$4,000	Cost of creating TLSA record for the certificate, similar to [SSK+18b]. Existing tools to automatically generate TLSA records (*https://ssl-tools.net/tlsa-generator*).

7.4.5.2 Other Mitigations

DNS-Based Authentication of Named Entities (DANE) [HS12] is a DNS-level mitigation against vulnerabilities in the CA model [OKL+12]. DANE allows to retrieve an end-entity certificate or a certificate to be found in the path to (including) the trust anchor through DNS queries. Depending on the implementation, DANE can amend or side step the CA model; we consider the case where DANE defines the website's current end-entity certificate in its record. This mitigation requires DNSSEC to be deployed. Although the adoption of DANE for email servers is growing, there are challenges that prevent the adoption for the Web PKI [LGvR+20]. As of now all major browsers do not automatically validate DNSSEC and DANE. We nevertheless assumed that this feature is implemented and evaluated its effect as an alternative to CT in case of the presence of a non-stealthy attacker in Section 7.8.

Other mitigations like the DNS Certificate Authority Authorization (CAA) allows domain owners to specify via CAA records the CAs that are allowed to issue certificates for the domain. However, this does not prevent a malicious CA to issue certificates [HSH19]. We thus ignored this mitigation.

7.4.6 Mitigation Cost

We focus on the immediate cost of mitigations and convert all personnel cost from time estimates by considering the cost for an external consultant[8] of $r = 140\,\$/\mathrm{h}$. Table 7.2 lists the cost estimates.

7.4.7 Discussion

Like our threat model in Section 7.3, our mitigations inherently focus on protocol-level attacks. We assume application layer mitigations to be correctly implemented, which we try to capture by allocating sufficient cost to employ expert consultants. Naturally, there is still a probability of failure that depends on the web application, which we could in principle capture as probabilistic failure (see Part I), but needs additional empirical data. We *over-approximate* the efficacy of browser-level mitigations by assuming all users to use current browser versions. Our results thus have to be read either as a projection to the future (about the potential of these mitigation techniques) or as a security analysis for the share of users with recent browsers. This limitation can be overcome by determining which browser versions implement which mitigation and using per-website data on browser usage. We *approximate* the cost of mitigations by considering a uniform set of rules and assuming similar cost of labor in the web security sector. All our estimates consider only direct cost.

7.5 Detailed Web Threat and Defender Model

We detail the complete threat and defender model described in Section 7.3 and Section 7.4 here. If you want to skip this, you can continue reading Section 7.6 on page 133.

This section contains the complete model describing JavaScript-based attacks. Each predicate in the *precondition* of an action is in conjunction with the other predicates; the presence of disjunctions in a rule is used as shorthand to represent different rules for the same action with a shared part in the *precondition*.

Table 7.3 contains the entire list of predicates used in the model. Using the notation of the Boolean Logic, the symbol ¬ in front of a predicate negates the predicate itself.

7.5.1 Attacker Propagation Rules

This section describes the propagation rules for the attacker used in the threat model. We provide each rule, followed by the intuition of what kind of attack it represents.

7.5.1.1 Initially Compromised Nodes

$$\frac{\substack{x \in AS \cup IP \cup NS \cup CA \\ cn \in Country \quad x \xrightarrow{\text{loc}} cn \quad C(cn)}}{C(x)} \tag{1}$$

[8] Hourly rate (US) for a Computer Security System Specialist level 2 via Deloitte, Ltd [Del].

Table 7.3: Threat model predicates.

Predicate	Description
$x \xrightarrow{\text{loc}} cn$	The element $x \in AS \cup IP \cup D \cup NS$ is located in $cn \in Country$
$d \xrightarrow{\text{A}} i$	The element $d \in D \cup NS$ has address $i \in IP$
$i \xrightarrow{\text{orig}} a$	The element $i \in IP$ belongs to $a \in AS$
$c \xrightarrow{\text{JS}} d$	The element $d \in D$ contains JS scripts hosted in the element $c \in D$
$avail\_over\_HTTPS(c, d)$	The JS resources retrieved from c by d are available over HTTPS
$e \xrightarrow{\text{DNS}} d$	The element $e \in NS$ is one of the authoritative name servers of d
$p \xrightarrow{parent\_zone} e$	The element $p \in NS$ manages the parent zone of the element $e \in NS$
$a \xrightarrow{\text{RTE(b)}} c$	Given $a, b, c \in AS$, the route from a to c passes through b
$C(x)$	The element $x \in AS \cup IP \cup D \cup Country \cup NS \cup CA$ is compromised. In case $x \in D \cup NS$, the predicate means that x can be used to distribute malicious JavaScript. This predicate will be called *Globally compromised*
$C^{\text{web}}(d)$	The website hosted on $d \in D$ is compromised. JS included from d is not necessarily compromised as well. This predicate will be called *Website access compromised*
$C^{\text{web}}(c, d)$	The website on $d \in D$ is considered compromised for all the visitors from $c \in Country$. This predicate will be called *Website access compromised from*
$XSS(d)$	The element $d \in D$ is vulnerable to XSS
$UpgradeRequests(d)$	The element $d \in D$ employs the field upgrade-insecure-requests in the CSP to force HTTPS for all the resource requests.
$SRI(d, c)$	The element $d \in D$ implements the Sub-Resource Integrity mitigation for **all** the resources, used by d, stored in $c \in D$. It is assumed $d \neq c$
$IPsec(a, b)$	The packets routed between $a \in AS$ and $b \in AS$ are protected via IPsec
$DNSSEC(f)$	The element $f \in NS$ implements DNSSEC
$HTTPS(d)$	The element $d \in D$ implements HTTPS
$l\_HTTPS(d, e)$	**All** the JS resources, used by $d \in D$ and hosted in $e \in D$, are explicitly using HTTPS in the source code
$l\_HTTPS\_compat(d, e)$	**All** the JS resources, used by $d \in D$ and hosted in $e \in D$, are either explicitly using HTTPS in the source code or a protocol-relative URL
$HSTS(d)$	The element $d \in D$ implements the header strict-transport-security
$Redirect(d)$	The element $d \in D$ redirects HTTP connections to HTTPS. The redirection is either temporary (status code 302) or permanent (status code 301)
$CT(d)$	The digital certificates, for the element $d \in D$, are signed by CAs that are compliant with the Certificate Transparency
$DANE(d)$	The element $d \in NS$ implements DANE
$I^{\text{DNS}}(d)$	The DNS resolution of the element $d \in D$ is compromised. This predicate will be called *Globally compromised DNS*
$I^{\text{DNS}}(d, e)$	The DNS resolution of the element $d \in D$ is compromised for the visitors from $c \in Country$. This predicate will be called *Country compromised DNS*
$I^{\text{R}}(i, j)$	The route between $i, j \in IP$ is compromised
$I^{\text{CA}}(d)$	The element $d \in D$ is vulnerable to certificate authority attacks
$TLSA\_0(a)$	The element $a \in CA$ is present in the certificate chain of the TLSA record with certificate usage field 0
$TLSA\_2(a)$	The element $a \in CA$ is present in the certificate chain of the TLSA record with certificate usage field 2

Intuition: All the autonomous systems, IPs, name servers and certificate authorities associated to a malicious country are under the control of the attacker.

$$\frac{i \in IP \quad a \in AS \quad i \xrightarrow{\text{orig}} a \quad C(a)}{C(i)} \tag{2}$$

Intuition: All the IPs, that belong to an autonomous system compromised by the attacker, are considered under the control of the attacker.

$$\frac{i \in IP \quad d \in D \cup NS \quad d \xrightarrow{\text{A}} i \quad C(i)}{C(d)} \tag{3}$$

Intuition: If a domain (name server) resolves to an IP address under the control of the attacker, then also the domain (name server) is considered compromised.

$$\frac{i \in IP \quad d \in D \cup NS \quad d \xrightarrow{\text{A}} i \quad C(d)}{C(i)} \tag{4}$$

Intuition: The same applies in the opposite direction. If a domain or NS is compromised, the corresponding IP is also considered compromised.

7.5.1.2 Content Compromise

$$\frac{d \in D \quad XSS(d)}{C^{\text{web}}(d)} \tag{5}$$

Intuition: If a web server is vulnerable to XSS attacks then the attacker can gain control of the content of the website. We did not consider using a Content Security Policy because its impact on the functionality of a website is currently not measurable. For example, CDNs often inject scripts in websites and thus the cost of deploying a CSP can hardly be measured [SMJ+21].

7.5.1.3 DNS Compromise

$$\frac{d \in D \quad e \in NS \quad e \xrightarrow{DNS} d \quad C(e)}{I^{\text{DNS}}(d)} \tag{6}$$

Intuition: If one of the authoritative name servers of a domain is under the control of the attacker, then the DNS resolution for this domain is considered compromised[9]. An attacker can modify the DNS resolution and map the domain name to a different IP.

[9] Due to the fact that there is no information about which authoritative NS is queried by a client, this is a simplification implemented in the model. Furthermore, if the attacker is able to compromise one of the authoritative NS for a domain, it is possible that it is also able to compromise the other NSs.

7.5.1.4 Route Compromise

$$\frac{\substack{a,b,c \in AS \quad a \neq b \neq c \quad C(b) \\ a \xrightarrow{RTE(b)} c \quad \neg IPsec(a,c) \quad i \xrightarrow{orig} a \quad j \xrightarrow{orig} c}}{I^{\mathrm{R}}(i,j)} \tag{7}$$

Intuition: If a route from one AS to another AS is not protected via IPsec and it passes through a third AS under the control of the attacker, then the route is insecure and the two endpoints of the communication could be targeted by an attack. This rule does not consider the case in which the sender or the destination is compromised, because IPsec cannot protect against this scenario.

$$\frac{a \in AS \quad C(a) \quad i \in IP \quad j \in IP \quad i \xrightarrow{orig} a}{I^{\mathrm{R}}(i,j)} \tag{8}$$

Intuition: If the sender AS is under the control of the attacker, then **all** the routes originating from this AS are deemed to be insecure. This scenario describes the situation where a country or a provider implements surveillance over its population or customers. Note that the case in which an endpoint is compromised, e.g., j in this rule, is captured by (2), which would mark the respective IP and thus domain or name server compromised.

7.5.1.5 Route to Web Server Compromise

$$\frac{\substack{e \in Country \quad d \in D \quad a \in AS \quad i,j \in IP \\ d \xrightarrow{\mathrm{A}} j \quad i \xrightarrow{orig} a \quad a \xrightarrow{loc} e \quad I^{\mathrm{R}}(i,j) \\ \neg(HTTPS(d) \wedge \neg I^{\mathrm{CA}}(d) \wedge Redirect(d) \wedge HSTS(d))}}{C^{\mathrm{web}}(e,d)} \tag{9}$$

Intuition: This threat model assumes the worst scenario in which a *non-tech-savvy* user accesses the web server via HTTP. It is important to underline that HTTP is the default protocol used by browsers if a protocol is not explicitly defined. If a route between a client and a web server is insecure, then the attacker can implement a MITM attack in the following cases:

- *Case 1:* If the web server does not implement HTTPS, then the attacker can eavesdrop and replace the content retrieved from the web server.
- *Case 2:* If the web server implements HTTPS but it does not redirect to HTTPS, then, for the hypothesis previously presented, the attacker can eavesdrop and replace the content retrieved from the web server. The HSTS header does not provide any protection if Redirect is not implemented; indeed the header is ignored in an HTTP connection [HJB12].
- *Case 3:* If the web server implements HTTPS and redirects HTTP traffic to HTTPS but it does not implement HSTS, then the attacker can compromise the connection before the redirection phase.

- *Case 4:* If the web server implements HTTPS but is vulnerable to certificate authority attacks[10], then a malicious CA can forge digital certificates for the domain and use them to authenticate connections to malicious web servers.

Case 3 does not take into account the fact that, using a permanent redirection, the new URI is cached; therefore all the new requests for the resource will be automatically mapped to the new URI [FR14]. This model additionally requires the presence of the strict-transport-security header. This choice is due to the fact that Redirect is not a secure mitigation and HSTS provides a better security with respect to the permanent redirection:

- HSTS covers the entire domain.
- HSTS implements a preloaded list[11].

For those domains that are not in the preloaded HSTS list, the first access to a web server is still insecure even if all the previous requirements are met.[12] The attacker in our model is not allowed to exploit this vulnerable window to implement a MITM attack; the attacker can compromise the subsequent connections.

The *postcondition* of this rule declares that all the connections originating from the country where the sender AS is located, are compromised. This is an upper bound assumption because it is possible that there exist ASes, located in the country, that do not present an insecure route. This simplification is due to the fact that there is no information about the location within the country of the client contacting the web server.

7.5.1.6 Route to Name Server Compromise

$$\frac{\begin{array}{c} a{\in}AS \quad i{,}j{\in}IP \quad e{\in}Country \quad a\xrightarrow{loc}e \quad i\xrightarrow{orig}a \\ f\xrightarrow{DNS}d \quad f\xrightarrow{A}j \quad I^{\mathrm{R}}(i{,}j) \quad \neg DNSSEC(f) \end{array}}{I^{\mathrm{DNS}}(d,e)} \tag{10}$$

Intuition: If a route between a client and a name server is insecure and the NS does not implement the DNSSEC protocol, then the attacker can redirect the client to a malicious NS or can implement a DNS cache poisoning attack. As a result, the DNS resolution of the queried domain is compromised for all connections originating in the country where the client AS is located[13].

7.5.1.7 From DNS to Domain Compromise

$$\frac{I^{\mathrm{DNS}}(d) \quad \neg(HTTPS(d)\wedge\neg I^{\mathrm{CA}}(d)\wedge Redirect(d)\wedge HSTS(d))}{C^{\mathrm{web}}(d)} \tag{11}$$

[10] See rules: 16, 17, 18

[11] It is a list of domains which are automatically configured with HSTS. This list is integrated in the browser.

[12] A possible mitigation for this scenario is to increase the number of domains contained in the preloaded HSTS list.

[13] This is the same simplification presented in rule 9

Intuition: If a web server has a *Globally compromised DNS*[14] and either does not fulfill all the conditions to establish a secure connection[15] or the attacker is able to forge a malicious certificate for the website, then the attacker can redirect **all** clients to a malicious web server that is able to claim to be the legitimate one.

$$\frac{\begin{array}{c} I^{\text{DNS}}(d,e) \quad e{\in}Country \\ \neg(HTTPS(d){\wedge}\neg I^{\text{CA}}(d){\wedge}Redirect(d){\wedge}HSTS(d)) \end{array}}{C^{\text{web}}(e,d)} \tag{12}$$

Intuition: The same situation applies in case the web server has a *Country compromised DNS*; the only difference lies in the *postcondition*, where the attacker can only redirect the clients from the particular country to a malicious web server.

$$\frac{\begin{array}{c} I^{\text{DNS}}(c) \quad c\xrightarrow{JS}d \quad \neg SRI(d,c) \\ \neg(HTTPS(d){\wedge}Redirect(d)){\vee}I^{\text{CA}}(c) \\ \neg l\_HTTPS(d,c){\vee}I^{\text{CA}}(c) \quad \neg\,UpgradeRequests(d){\vee}I^{\text{CA}}(c) \end{array}}{C^{\text{web}}(d)} \tag{13}$$

Intuition: If a CDN, that provides JS resources for a certain web server, has a *Globally compromised DNS*, then the attacker can redirect the client to a CDN that provides malicious JS resources. This scenario is possible if **all** the following conditions are met:

- The web server does not implement the Subresource integrity mitigation, thus the JS resource can be replaced with a malicious one.
- The protocol used to retrieve the resource from c is not HTTPS or the attacker is able to forge a malicious certificate for the CDN.
- The web server does not implement the upgrade-insecure-requests field in the CSP or the attacker is able to forge a malicious certificate for the CDN.
- The website is not accessible via HTTPS or does not redirect automatically to the secure protocol or the attacker is able to forge a malicious certificate for the CDN.

Note that the website on d is required to implement a redirect to HTTPS only if it uses protocol-relative URLs. In case d does deliver its content via HTTP and includes resources explicitly via HTTPS, it is able to protect against this attack on the resolution of c.

$$\frac{\begin{array}{c} I^{\text{DNS}}(c,e) \quad e{\in}Country \quad c\xrightarrow{JS}d \quad \neg SRI(d,c) \\ \neg(HTTPS(d){\wedge}Redirect(d)){\vee}I^{\text{CA}}(c) \\ \neg l\_HTTPS(d,c){\vee}I^{\text{CA}}(c) \quad \neg\,UpgradeRequests(d){\vee}I^{\text{CA}}(c) \end{array}}{C^{\text{web}}(e,d)} \tag{14}$$

Intuition: The same situation applies in case the CDN has a *Country compromised DNS*; the *postcondition* presents a similar structure as rule 12.

[14] This means that the attacker has control over the content provided by one of the authoritative NSs for this domain.

[15] A secure connection via HTTPS allows to authenticate the endpoints.

7.5.1.8 Inline JS Injection

$$\frac{\begin{array}{c} c{\in}Country \quad i,j{\in}IP \quad d_1,d_2{\in}D \quad d_2 \xrightarrow{JS} d_1 \quad a{\in}AS \\ i \xrightarrow{orig} a \quad d_2 \xrightarrow{A} j \quad I^{\mathrm{R}}(i,j) \quad a \xrightarrow{loc} c \quad \neg SRI(d_1,d_2) \\ \neg(HTTPS(d_1) \wedge Redirect(d_1)) \vee I^{\mathrm{CA}}(d_2) \\ \neg l\_HTTPS(d_1,d_2) \vee I^{\mathrm{CA}}(d_2) \quad \neg UpgradeRequests(d_1) \vee I^{\mathrm{CA}}(d_2) \end{array}}{C^{\mathrm{web}}(c,d_1)} \tag{15}$$

Intuition: If the route from a client to a CDN, which provides JS resources to a web server, is insecure[16] and **all** the following conditions are met:

- The web server does not implement the Subresource integrity mitigation: in this case a MITM attacker can drop the legitimate JS resource and can replace the content with malicious code.
- The protocol used to retrieve the resources of the CDN in the web server HTML code of d_1 is not HTTPS or the attacker is able to forge a malicious certificate for the CDN.
- The web server does not implement the upgrade-insecure-requests field in the CSP or the attacker is able to forge a malicious certificate for the CDN.
- The website is not accessible via HTTPS or does not redirect automatically to the secure protocol or the attacker is able to forge a malicious certificate for the CDN.

Then, the attacker can intercept the JS requests and inject malicious JS code. Note that we made sure that no mitigation (*UpgradeRequests* or *l_HTTPS*) breaks the functionality of the website by requiring that the resource is available over HTTPS (see Section 7.5.2).

7.5.1.9 Certificate Compromise

$$\frac{\begin{array}{c} a{\in}CA \quad d{\in}D \quad e{\in}NS \quad C(a) \\ e \xrightarrow{DNS} d \quad \neg CT(d) \quad \neg DANE(e) \end{array}}{I^{\mathrm{CA}}(d)} \tag{16}$$

Intuition: If a certificate authority is under the control of the attacker and the following conditions are met:

- The web server's digital certificates are signed by CAs that are not compliant with the Certificate Transparency project.
- The authoritative NSs of the domain do not implement the DANE protocol.

Then, the attacker can forge malicious digital certificates for the domain and use them to generate authenticated connections to malicious web servers.

$$\frac{\begin{array}{c} a{\in}CA \quad d{\in}D \quad e{\in}NS \quad C(a) \\ e \xrightarrow{DNS} d \quad \neg CT(d) \quad DANE(e) \quad C(e) \end{array}}{I^{\mathrm{CA}}(d)} \tag{17}$$

[16] This model assumes that the web server does not implement a Proxy to retrieve the resources from the CDN on behalf of clients.

Intuition: If, in the same scenario of rule 16, one of the NS is under the control of the attacker, the DANE protocol cannot be trusted. For example, the attacker can modify the TLSA records and insert a new hash of a digital certificate signed by the compromised CA.

$$\frac{\substack{a{\in}CA \quad d{\in}D \quad e{\in}NS \quad C(a) \quad e\xrightarrow{DNS}d \\ \neg CT(d) \quad DANE(e) \quad \neg C(e) \quad (TLSA\_0(d,a){\vee}TLSA\_2(d,a))}}{I^{\mathrm{CA}}(d)} \tag{18}$$

Intuition: If, in the same scenario of rule 16, the authoritative NS is not compromised and implements the DANE protocol, the attacker can forge new digital certificates if one of these two conditions is met:

- The TLSA certificate usage field is 0 and the compromised CA is in the Certificate Chain.[17]
- The TLSA certificate usage field is 2 and the compromised CA is in the Certificate Chain from the Server certificate to the Trust anchor.

7.5.1.10 Third-Party JS Injection

$$\frac{d, e \in D \quad e \xrightarrow{\text{JS}} d \quad \neg SRI(d, e) \quad C(e)}{C^{\mathrm{web}}(d)} \tag{19}$$

Intuition: If a web server contains a JS resource that is not protected via Subresource Integrity and is hosted in a domain under the control of the attacker, then the attacker can replace the content of the JS script with malicious code.

7.5.1.11 Access Compromised to Website Access Compromised

$$\frac{d \in D \quad C(d)}{C^{\mathrm{web}}(d)} \tag{20}$$

Intuition: If a domain d is globally compromised, the website on d is compromised as well.

7.5.2 Defender Rules

This section describes the propagation rules for the defender used in the threat model. We describe only those rules that require some preconditions to be implemented. The remaining mitigations have no preconditions.

[17] The model assumes that the TLSA record defines the entire chain; this is the most secure approach.

7.5.2.1 Secure Inclusions

$$\frac{d, c \in D \quad c \xrightarrow{JS} d \quad avail\_over\_HTTPS(c, d)}{l\_HTTPS(d, c)} \tag{21}$$

Intuition: If a web server contains JS resources from a different domain which are **all** available over HTTPS, then the defender can explicitly enforce HTTPS for retrieving the JS resource in the source code. We do not allow new domains to use protocol-relative URLs (*l_HTTPS_compat*) because it is an anti-pattern and if resources are available over HTTPS they can always be retrieved explicitly over HTTPS even if the domain d is using HTTP.

$$\frac{d, c \in D \quad \bigwedge_{c.c \xrightarrow{JS} d} avail\_over\_HTTPS(c, d)}{UpgradeRequests(d)} \tag{22}$$

Intuition: If the entry *UpgradeRequests* of the CSP shall be utilized, we need to check that all the JS resources retrieved from all the different domains c are retrievable over HTTPS.

7.5.3 Redirection to HTTPS and HSTS

$$\frac{\begin{array}{c} d,c \in D \quad HTTPS(d) \\ (\bigwedge_{c.c \xrightarrow{JS} d} (l\_HTTPS(d,c) \vee (l\_HTTPS\_compat(d,c) \wedge avail\_over\_HTTPS(c,d)))) \\ \vee\, UpgradeRequests(d) \end{array}}{Redirect(d)} \tag{23}$$

Intuition: To implement a redirection over HTTPS to the domain d, it must implement HTTPS and all the included JS resources from external domains must be retrieved over HTTPS (either explicitly or using protocol-relative URLs). This is required to not break functionality of the domain d. Indeed, if a redirection over HTTPS is established, but the JS resources are not retrievable over HTTPS, it will trigger a mixed-content warning in all major browsers. In case the domain employs protocol-relative URLs (l_HTTPS_compat), it is also required to have the resource available over HTTPS. The predicates obtained from the rules 21 and 22 already required the availability of the resources over HTTPS.

$$\frac{d \in D \quad HTTPS(d)}{HSTS(d)} \tag{24}$$

Intuition: The precondition to implement the security header *HSTS* is the implementation of HTTPS on the domain.

7.5.4 DNSSEC

$$\frac{e, p \in NS \quad \bigwedge_{p.p \xrightarrow{parent\_zone} e} DNSSEC(p)}{DNSSEC(e)} \tag{25}$$

Intuition: The precondition to deploy *DNSSEC* on a name server is the implementation of DNSSEC in all the parent zones.

7.5.4.1 DANE

$$\frac{e \in NS \quad DNSSEC(e)}{DANE(e)} \tag{26}$$

Intuition: The precondition to deploy *DANE* on a name server is the implementation of DNSSEC.

7.5.5 Certificate Transparency

$$\frac{d \in D \quad HTTPS(d)}{CT(d)} \tag{27}$$

The precondition to employ Certificate Transparency logs is that the domain implements HTTPS, i.e., it has a digital certificate.

7.6 Implementation

We discuss data acquisition, pre-processing and how to solve the resulting Stackelberg problem.

7.6.1 Data Acquisition

Our mitigation analysis is performed on the Top 5k Alexa domains obtained from Tranco [PvGT$^+$19] on 1 Oct 2020. The data collected represents a snapshot of the status of the Internet at a specific moment. Out of the 5k domains, 4608 were accessible (92%) at the time of the crawl. The remaining domains either provide services not related to web browsing or were down. We crawled each accessible domain to collect the web server configuration, its CDNs, DNS data, routing data, geolocation, and (if applicable) CA information. The data collection was performed from a single location at a European university. We then identified a subset of domains with XSS vulnerabilities using taint tracking. [SRJ$^+$19]

7.6.1.1 Web Server Data

We collected the security headers strict-transport-security and Content-Security-Policy to determine the presence of the HSTS and CSP mitigation respectively. In the case of CSP, we parse for the presence of the *upgrade-insecure-requests* directive. We ignored the remaining policies.

We then probe the server with HTTP requests on the standard port 80 to collect the sequence and type of redirections to an HTTPS connection.

7.6.1.2 CDN JS Resources

For every domain name, we extract the external JS resources that are loaded either statically or dynamically, analyze the protocol and check for the presence of the subresource integrity[18]. As the content of the landing page and its internal pages likely differ [ACF$^+$20], to avoid the risk of capturing a limited subset of third-parties [UDH$^+$20], we further visited up to 25 random internal pages obtained from the links on the landing page of the visited domain. We employed the *tldextract* package [Pyt20] to extract the TLD+1 of each resource. We consider a resource an internal page that belongs to the domain visited if it shares the TLD+1 with the landing page of the domain but have a distinct URL by excluding the fragment component. For example, from the landing page of the domain *foo.com*, the URL *foo.com/#home* is not considered an internal page while the URLs *foo.com/content/index.html* and *bar.foo.com/index.html* are visited as internal pages.

7.6.1.3 DNS Data

For each website and CDN, we collect the list of authoritative name servers that are contacted during iterative DNS resolution. We then queried for DNSSEC and DANE records. For DANE, we requested the TLSA record for the website on port 443. For DNSSEC we required the DNSKEY records for the zone, and we then trace the presence of the DS and its RRSIG records in the parent zones up to the root zone. Although DNSSEC is prone to misconfiguration [CvRC$^+$17] (e.g., expired signatures), we did not investigate these issues.

7.6.1.4 Routing Information

To model the Internet connectivity, i.e.connectivity between ASes, we collect traceroutes provided by RIPE Atlas [RIP17a] for the set of autonomous systems considered in our dataset. We observed traceroutes that have the domains from our dataset as their destination. For each ASN, we then retrieved the holder name.

7.6.1.5 Geolocation

For each NS, website, and CDN, we include their geolocation in our dataset. We link IPs to ASes using the RIPEstat database service [RIP17b] and we map ASes to countries

[18] We used the *Selenium Web* driver, an object-oriented API for web-app testing

using the MaxMind database [Max17]. In addition, each CA is mapped to a specific country using the information stored in the issuer section of the digital certificate.

7.6.1.6 CAs

For websites that support HTTPS, we use the X.509 certificate to identify the issuing certification authority. To that end, we combine the information stored in *Certificate Fields* that are reserved for the issuer of the certificate: Common Name (CN), Organization (O) and Organizational Unit (OU). Finally, to identify the country, i.e.the administrative entity of CA, we also collect the Country (C) field from the certificate.

7.6.2 Limitations & Caveats

7.6.2.1 Visitors

To calculate the attacker reward (see Section 7.3), we collected statistics about the number of visits on each domain using the Alexa Web Information Service and we employed the UrlInfo API[19] to get the number of domain views out of a million total page views made on the Internet.

7.6.2.2 Routing Information

Achieving a global and complete routing coverage, where all possible routes between ASes are covered is an infeasible task, if not impossible [GIL$^+$12; OPW$^+$10]. BGP data is available to a limited extent, leaving a meaningful part of the AS-level topology hidden. However, to cover as many routes as possible, we collect routes that were created with the RIPE Atlas [RIP17a] networks at the beginning of 2021[20].

7.6.3 Stackelberg Planning via Graph Databases

We derived a general-purpose algorithm for Stackelberg planning in Chapter 3 which was successfully applied to the security of local computer networks in Chapter 5 and the email infrastructure in Chapter 6. The algorithm uses a diverse set of optimizations and pruning techniques to reuse information gathered across different mitigation scenarios and to discover when mitigations are applicable in no particular order. Di Tizio's Master thesis employed this algorithm for the web infrastructure [Tiz18] with a similar threat model. Despite using all available optimizations, experiments only scaled up to about 50 domains, exceeding the available memory of 88 Gb after several days of computation. This is due to the problem size: reading the input file and initializing internal data structures already takes hours, even though computing the attacker plan is simple once these structures are in place.

Hence we developed a new approach based on the *Neo4j* graph database system. Instead of enumerating all relevant attacker actions in an input file (like in [Tiz18]), we generate an attack graph that captures their relation and thus allows efficient computation of the attacker reward via reachability analysis and application of defender

[19] https://docs.aws.amazon.com/AlexaWebInfoService/latest/ApiReference_UrlInfoAction.html

[20] We assume that BGP routes are reasonably stable [RWX$^+$02].

actions via the removal of edges. Neo4j's data structures are optimized for such queries. Moreover, by representing the data presented in Section 7.6.1 as a property graph, we drastically improve the generation time of this attack graph. Concluding, while our algorithms from Part I work great for complex problems with reasonably small description (i.e., number of facts and actions), special purpose algorithms can be more suitable for huge but non-complex problems. We provide a formal overview of our graph-based analysis in the following section.

7.7 Graph-Based Analysis

We use Neo4j to analyze a larger set of domains. In contrast to the fine-grained deployment analysis via Stackelberg planning, which considered the best-possible mitigation *per host*, we consider a fixed set of mitigation scenarios. This is not necessarily optimal, as the optimal deployment can be a mix of two solutions. On the other hand, policy decisions often do not afford a per-host policy. Hence, our global policies are more realistic to be carried out.

7.7.1 Notation

In planning, the set of actions for a yet-to-be-specified problem is defined using so-called *action schemas*, or *rules*. In contrast to an action, such a schema can contain variables over some fixed domains in the postconditions $\text{post}(r)$ and preconditions $pre(r)$. Our rules (cf. Section 7.5) slightly extend this notation. Variables are defined over the nodes of a property graph $\text{PG} = (V, E)$. In addition to precondition and postcondition, a rule r has a graph property $\text{graph}(r)$ which is a conjunction of atoms '$v \in S$' (v is in a named subset $S \subseteq V$) and '$v_1 \xrightarrow{S} v_2$' (v_1, v_2 are connected with an edge in $S \in E$). Using Neo4J, we can efficiently evaluate these properties and find all satisfying assignments from variables to nodes in the graph. We say that an assignment $\sigma : \mathcal{V} \to V$ satisfies a graph constraint ϕ on a property graph $\text{PG} = (V, E)$, written $\text{PG}, \sigma \vdash \phi$ iff

$$\begin{aligned}
&\text{PG}, \sigma \vdash x \in S && \iff x\sigma \in S \subseteq E \\
&\text{PG}, \sigma \vdash x \xrightarrow{z} y && \iff (x\sigma, y\sigma) \in S \subseteq E \\
&\text{PG}, \sigma \vdash \phi_1 \wedge \phi_2 && \iff \text{PG}, \sigma \vdash \phi_1 \text{ and } \text{PG}, \sigma \vdash \phi_2
\end{aligned}$$

7.7.2 Rule Dependencies

To exploit Neo4J's strengths, we need to minimize the number of queries and computations outside the query evaluation. We exploit the structure of our threat model to this end. First, we observe that all rules r have only a single postcondition. We can relate our rules in a dependency graph (Figure 7.2). Nodes are conditions, i.e., predicates with variables. Two nodes are connected if there is a rule with the first node as precondition and the second as postcondition. Only rules (17) and (18) have two preconditions, which we indicate with the $\wedge$ symbol.

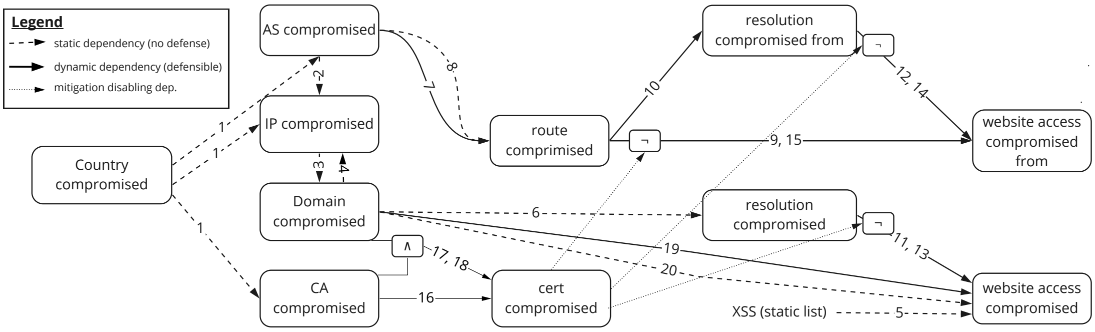


Figure 7.2: Dependency graph: hierarchy of rules and compromise predicates.

Second, the dependency graph reveals that there is only a single loop (rules (3), (4)) in this graph. Apart from this loop, every predicate can be derived with a bounded number of steps that corresponds to the length of the equivalent path in our dependency graph. This allows us to express the attacker search with a bounded number of Neo4j queries that generate all predicates in the final state.

The number of applications of (3) and (4) is unbounded in general, but in practice (and on our dataset) this fixpoint computation finishes after three steps. Disregarding the loop, we can use any topological order of the dependency graph to iteratively build the corresponding attack graph AG. At any step, we add the postconditions of the current rule r given that all possible instances of its preconditions are already present in AG and that the graph conditions can be evaluated on PG. The loop ((3) and (4)) is handled separately.[21] In the resulting graph AG, a node represents an instantiation of a predicate and an edge represents an instantiation of a rule.

7.7.3 Attack Graph Generation

We generate one AG per attack scenario. By fixing the attack scenario, we can compute the effect of mitigations as a simple removal of edges and a reachability query in Neo4J. Note that the number of removed edges for a mitigation is often relatively small compared to the size of the attack graph.

Algorithm 7.1 is used to translate a property graph into an attack graph. It applies to all threat models that can be described with a dependency graph, i.e., have a single positive postcondition and where mitigations only disable, but never enable attacker actions. It can handle loops, but is most efficient if they concern only a small number of nodes in the dependency graph. Starting from the initial country compromise rules (line 1), it traverses every rule in topological order w.r.t. the dependency graph (lines 2 and 7). For each rule, it formulates a query that generates the set of nodes (= compromise predicates) and edges (=actions) that represent applications of this rule valid for the property graph. Lines 4-6 handle the loop consisting of (3) and (4). The

[21] For the general case of any dependency graph: We first compute all strongly connected components (SCCs) of the graph and replace any SSC with more than one node by a single placeholder node. Second, we sort the graph consisting of all the (placeholder) nodes and process them according to this order. If we encounter a placeholder, we perform the fixpoint computation of all the nodes which were originally replaced by the placeholder.

Algorithm 7.1: Property graph to attack graph.

```
   Input: property graph PG, compromised countries Cntry
   Output: attack graph AG
   // initialize AG
1  AG ← $(V, \emptyset)$ with $V = \{C(cn) \mid cn \in \text{Cntry}\}$;
   // add compromised nodes
2  for $r_i \in [1, 2]$ do
3      AG ← AG + $\{(pre(r)\sigma \rightarrow post(r)\sigma) \mid \text{PG}, \sigma \vdash \text{graph}(r_i)\}$;
   // apply (3) and (4) until fixpoint is reached
4  while fixpoint not reached do
5      AG ← AG + $\{(pre(r)\sigma \rightarrow post(r)\sigma) \mid \text{PG}, \sigma \vdash \text{graph}(r_3)\}$;
6      AG ← AG + $\{(pre(r)\sigma \rightarrow post(r)\sigma) \mid \text{PG}, \sigma \vdash \text{graph}(r_4)\}$;
   // iteratively build graph
7  for $r_i \in [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]$ do
8      AG ← AG + $\{(pre(r)\sigma \rightarrow post(r)\sigma) \mid \text{PG}, \sigma \vdash \text{graph}(r_i)\}$;
   // add mitigations
9  mark all edges in AG as removable if a defender rule applies to it;
10 for ca ∈{compromised CAs in AG} do
11     if ca not reachable for attacker then
12         mark all rules {9, 15}, {12,14}, and {11,13} depending on ca as removable;
```

resulting graph AG after line 8 contains all attacker plans as paths starting from some ‘Country-compromised’ node.

7.7.4 Mitigation Analysis

While the generation of the attack graph can be slow (several minutes), it allows a rapid computation of attacker success in given mitigation scenarios (order of seconds). As each edge in the attack graph corresponds to a rule, and mitigation predicates appear only in negated form, the application of a mitigation corresponds to the removal of an edge in the attack graph.

For efficiency reasons, we apply mitigations in bulk, i.e., DNSSEC to all domains where it is both applicable and useful in removing edges. Let M be a set of mitigations (e.g., consisting of DNSSEC). To determine the cost and efficacy of applying M wherever possible, we query all edges in AG corresponding to rules which are disabled by an action in M and remove these edge. We say that a rule r is disabled by a mitigation m if the precondition of r includes the effect of m in negated form. We compute the cost of M by multiplying the number of domains for which we enabled DNSSEC with the cost of DNSSEC. The computation of the remaining attacker success is just a reachability query.

Using *transactions*, Neo4j permits us to store the unmodified attack graph, remove edges, and unroll this transaction later to reestablish the unmodified attack graph quickly.

The advantage of this approach is the fast computation of mitigation cost and attacker reward for a single set M. The downside is that for n classes of mitigations, we need to consider all 2^n combinations. This can be feasible for small n (e.g., for our case,

$n = 8$). By contrast, classical Stackelberg planning computes the best options *for each host* instead of the best global policy, where, n additionally scales with the size of the attack graph.

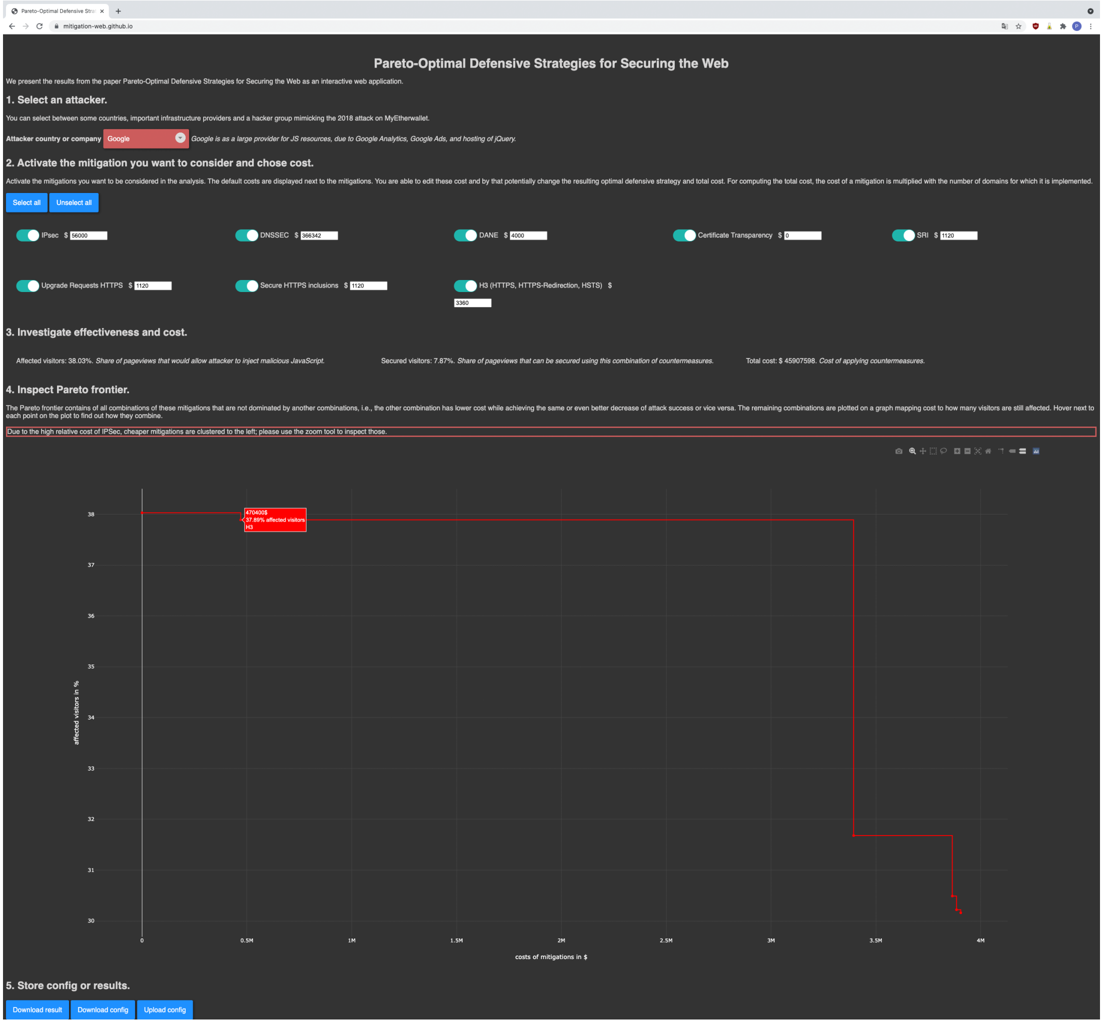
The only mitigation-disabling predicate that cannot be statically computed, i.e., based on PG, is the compromise of a certificate. As soon as the mitigations are known, however, the certificate compromise can be determined. There are only 466 CAs, distributed over various countries, hence we can thus afford to compute all compromised CAs (for simplicity), and then determine which of the attacker rules (9),(15), (12),(14) or (11),(13) are being disabled. They are disabled (marked with the ¬-symbol in Figure 7.2) if the corresponding mitigation was selected and the 'CA compromised' predicate preventing the mitigation is not reachable for the attacker.

7.8 Evaluation

We represented the data collected in Section 7.6.1 as a property graph and stored it in a Neo4j database. The size is around 8 Gb. we evaluated attacks on the Web carried by the different classes of attackers (a cyber-criminal group, large infrastructure providers offering, e.g., cloud services or name resolution, and nation-state groups) and the impact that the mitigations have in securing visitors on the Web. We model the purported threat by defining the set of assets initially under attacker control ((1)-(3) in Table 7.1).

For each attack scenario, we generated the attack graph. We then ran the analysis on every combination of the mitigation strategies introduced in Section 7.4. We computed from this: the impact of the attacker, in terms of % of visits in the Top 5k that can be affected by the attack vectors in the status quo, the *current efficacy*, in terms of % of visits in the Top 5k protected by the mitigations *currently* deployed, the *potential efficacy*, in terms of % of visits that could be protected by the application of different mitigations strategies (without breaking websites' functionality) globally on the Web, and the cost of applying these strategies to the status quo. In Table 7.4 and Table 7.5, we report this data for all sets of mitigation strategies we deem interesting – the totality of all 512 combinations would exhaust the space available here. For each scenario, we combined the set of combinations in a Pareto frontier. We removed each combination that is dominated by others (Section 7.2) and plotted the remaining ones on a graph mapping cost to potential efficacy.

All these computations, including the Pareto frontiers, can be interactively explored at mitigation-web.github.io(see Figure 7.3). Moreover, the user can modify the cost assigned to all countermeasures. Despite our best efforts to justify our cost assumptions, the empirical data available is incomplete and what needs to be taken into account is debatable. However, we can precompute each countermeasure's effect while also counting how often it is applied. The overall cost is the sum of these counts weighted by their cost and can be computed on the fly. The computation of the Pareto frontier is linear in the list of combinations once they are sorted by their cost.



Pareto-Optimal Defensive Strategies for Securing the Web

We present the results from the paper Pareto-Optimal Defensive Strategies for Securing the Web as an interactive web application.

1. Select an attacker.

You can select between some countries, important infrastructure providers and a hacker group mimicking the 2018 attack on MyEtherwallet.

Attacker country or company: Google *Google is as a large provider for JS resources, due to Google Analytics, Google Ads, and hosting of jQuery.*

2. Activate the mitigation you want to consider and chose cost.

Activate the mitigations you want to be considered in the analysis. The default costs are displayed next to the mitigations. You are able to edit these cost and by that potentially change the resulting optimal defensive strategy and total cost. For computing the total cost, the cost of a mitigation is multiplied with the number of domains for which it is implemented.

IPsec $ 56000
DNSSEC $ 366342
DANE $ 4000
Certificate Transparency $ 0
SRI $ 1120
Upgrade Requests HTTPS $ 1120
Secure HTTPS inclusions $ 1120
H3 (HTTPS, HTTPS-Redirection, HSTS) $ 3360

3. Investigate effectiveness and cost.

Affected visitors: 38.03%. *Share of pageviews that would allow attacker to inject malicious JavaScript.*

Secured visitors: 7.87%. *Share of pageviews that can be secured using this combination of countermeasures.*

Total cost: $ 45907598. *Cost of applying countermeasures.*

4. Inspect Pareto frontier.

The Pareto frontier contains of all combinations of these mitigations that are not dominated by another combinations, i.e., the other combination has lower cost while achieving the same or even better decrease of attack success or vice versa. The remaining combinations are plotted on a graph mapping cost to how many visitors are still affected. Hover next to each point on the plot to find out how they combine.

Due to the high relative cost of IPSec, cheaper mitigations are clustered to the left; please use the zoom tool to inspect those.

5. Store config or results.

Figure 7.3: Screenshot of mitigation-web.github.io.

Table 7.4: Percentage of affected visits, protected visits, and potentially protected visits and cost for infrastructure adversaries attacking the Alexa Top 5K. H3 is short for HTTPS, HTTPS-Redirect and HSTS, UR is short for CSP's upgrade-insecure-requests.

	companies (as attackers)									
Metric	Google		Amazon		GoDaddy		CloudFlare		Dyn	
Affected visits in status quo	38.03%		16.55%		12.3%		10.21%		7.62%	
Current efficacy in status quo (% of visits protected by the mitigations currently deployed)										
H3	0.05 %		0.05 %		0.05 %		3.15 %		4.46 %	
H3, SRI	0.05 %		0.08 %		0.05 %		3.15 %		4.46 %	
H3, UR	0.05 %		0.05 %		0.05 %		3.15 %		4.46 %	
CT	0.05 %		0.05 %		0.05 %		0.00 %		0.00 %	
CT, H3	0.05 %		0.05 %		0.05 %		3.15 %		4.46 %	
Potential efficacy (% of visits that could be protected by the mitigations) and deployment cost in $1000										
IPsec	0.00 %	2,072 k$	0.00 %	0 k$	0.00 %	4,424 k$	0.00 %	5,992 k$	0.00 %	0 k$
DNSSEC	0.00 %	39,931 k$	0.00 %	40,297 k$	0.00 %	41,030 k$	0.00 %	40,297 k$	0.00 %	39,931 k$
DANE	0.00 %	740 k$	0.00 %	740 k$	0.00 %	740 k$	0.00 %	0 k$	0.00 %	0 k$
SRI	6.35 %	3,393 k$	6.30 %	3,450 k$	4.21 %	440 k$	1.85 %	2,654 k$	0.00 %	42 k$
Sec. Incl.	0.00 %	10 k$	0.00 %	14 k$	0.00 %	3 k$	0.00 %	64 k$	0.00 %	38 k$
UR	0.00 %	5 k$	0.00 %	7 k$	0.00 %	3 k$	0.00 %	45 k$	0.00 %	11 k$
H3	0.15 %	470 k$	0.05 %	2,133 k$	0.17 %	221 k$	6.63 %	2,486 k$	6.49 %	272 k$
H3, CT	2.42 %	7,909 k$	9.01 %	8,880 k$	11.11 %	1,196 k$	6.63 %	2,486 k$	6.49 %	272 k$
H3, CT, SRI	7.60 %	3,884 k$	12.56 %	5,638 k$	11.18 %	685 k$	8.73 %	5,140 k$	6.49 %	314 k$
H3, CT, UR	2.69 %	8,430 k$	9.30 %	9,424 k$	11.40 %	1,263 k$	6.69 %	2,691 k$	6.76 %	309 k$
H3, CT, SRI, UR	7.87 %	3,904 k$	12.84 %	5,802 k$	11.44 %	705 k$	8.79 %	5,325 k$	6.76 %	328 k$

7.8.1 Infrastructure Providers

We first analyze the potential threat that the centralization of infrastructure in the Internet can pose to users in case an adversary gains control over them, and how to mitigate a potential attack. We choose some of the biggest infrastructure providers: Google, as a large provider for JS resources; CloudFlare and Amazon as two of the largest CDNs; Dyn, as one of the largest providers for DNS services and GoDaddy, as the largest domain registrar. Amazon, Google and GoDaddy also control certification authorities that are accepted trust anchors in all major browsers. The overall attacker success for each of the companies is shown in Table 7.4. Figure 7.4 shows the Pareto frontier for each company. The frontiers are a visualization of all Pareto-optimal combinations of mitigation strategy costs on the x-axis and the percentage of still affected visitors on the y-axis. As we can see, infrastructure providers have power similar to countries, with the largest one, Google, affecting about 38% of the page views versus the US controlling 46% (see following section). While Google's impact can only be reduced to 30% for relatively high cost, Amazon's impact can be reduced from 17% to 4% for even higher cost. In terms of attack vectors, we observed that Google has a great impact on routing. As expected, Amazon is able to compromise 3rd party resources either directly or via DNS spoofing. CloudFlare's major attack vectors are routing attacks and content compromise. Similarly, GoDaddy can exploit routing attacks on JS inclusions and name resolution, while Dyn's major attack vector relies on DNS poisoning. CloudFlare and Dyn control ASes, but no CA. While Dyn has 175 NSs serving about 576 domains in our dataset, it has little impact on them compared to Google and Amazon with roughly 8% of the visits are affected.

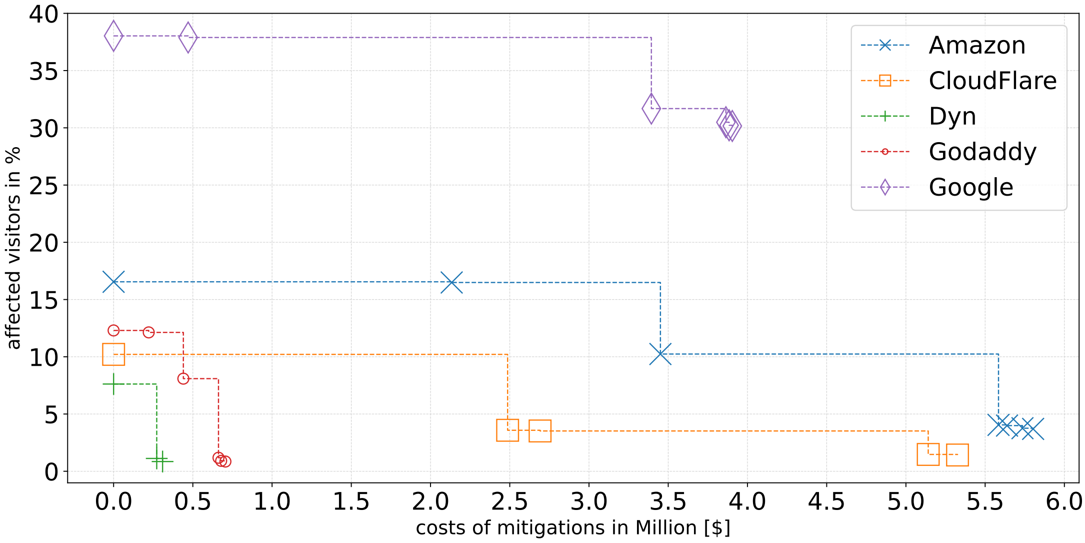


Figure 7.4: Pareto frontiers for infrastructure attacker scenarios.

The efficacy of currently deployed mitigations on securing visits is marginal (0.05% for Google and Amazon), showing that the current deployment is insufficient. However, if we apply a set of defenses globally, the potentially protected visits on the Top 5K increases to almost 8% for Google and 13% for Amazon.

Across the board (Table 7.4), we see that the deployment of lower-layer mitigations like IPsec, DNSSEC or DANE would add no additional security by themselves, even though they are often applicable, which is indicated by non-zero cost values. For Google, Amazon and GoDaddy, we see that SRI has a tremendous effect, as those host or control access to popular JS libraries, e.g., jQuery. This effect is less pronounced for CloudFlare and zero for Dyn, as these exert less control via JS inclusion and, specifically for Dyn, HTTPS is already protecting a great deal of connections.

H3+CT gives an inverse picture; the effect on Google is much weaker than SRI. It is important to underline that H3 deploys HTTPS, but does not assume CT compliance. As now all CAs support CT, H3+CT is the more realistic mitigation, deploying CT at zero cost. As expected, H3 has only very small impact in scenarios where the attacker controls a CA, highlighting the continued benefit of CT.

We observed that H3 is always the first points in the Pareto frontier, confirming the intuition that securing access to the first party comes at lower cost than protecting against JS inclusions. Securing third-party resources always achieves a stark improvement of security when combined with H3+CT, but comes at high cost. Whether SRI, Secure inclusions or CSPs upgrade-insecure-requests are cost efficient depends on how websites include third-party resources and whether they are controlled by the attacker. For Dyn and CloudFlare, UR is the best choice, as the direct compromise of the third-party is less of an issue. By contrast, SRI *by itself* appears in the Pareto frontiers for Google, Amazon, and GoDaddy due to their direct control of CDNs.

In all cases but Dyn, combining H3, SRI and UR achieves an increase over only H3 and SRI. This is because UR enables the deployment of H3 on domains with insecure

JS inclusions, where a secure redirect would otherwise break functionality.

Table 7.5: Percentage of affected visits, protected visits, and potentially protected visits and cost for hacker group and nation-state adversaries attacking the Alexa Top 5K. H3 is short for HTTPS, HTTPS-Redirect and HSTS, UR is short for CSP's upgrade-insecure-requests.

	hacker group		countries (as attackers)					
Metric	MyEtherWallet		US		CN		GB	
Affected visits in status quo	2.04%		46.01%		18.64%		13.93%	
Current efficacy in status quo (% of visits protected by the mitigations currently deployed)								
H3	1.55 %		0.00 %		0.26 %		0.00 %	
H3, SRI	1.55 %		0.00 %		0.26 %		0.00 %	
H3, UR	1.55 %		0.00 %		0.26 %		0.00 %	
CT	0.00 %		0.00 %		0.26 %		0.00 %	
CT, H3	1.55 %		0.00 %		0.26 %		0.00 %	
Potential efficacy (% of visits that could be protected by the mitigations) and deployment cost in $1000								
IPsec	0.00 %	0 k$	1.45 %	1,174,600 k$	9.43 %	216,104 k$	11.92 %	84,504 k$
DNSSEC	0.00 %	0 k$	0.01 %	52,386 k$	0.00 %	37,000 k$	0.00 %	12,455 k$
DANE	0.00 %	0 k$	0.00 %	740 k$	0.00 %	740 k$	0.00 %	740 k$
SRI	0.00 %	69 k$	5.46 %	4,331 k$	3.38 %	730 k$	5.15 %	590 k$
Sec. Incl.	0.00 %	67 k$	0.00 %	53 k$	0.00 %	23 k$	0.00 %	3 k$
UR	0.00 %	24 k$	0.00 %	35 k$	0.00 %	4 k$	0.00 %	3 k$
H3	1.18 %	1,827 k$	0.04 %	5,923 k$	0.15 %	544 k$	0.05 %	168 k$
H3, CT	1.18 %	1,827 k$	1.62 %	11,538 k$	9.53 %	1,515 k$	12.53 %	641 k$
H3, CT, SRI	1.18 %	1,897 k$	8.15 %	10,419 k$	9.81 %	1,398 k$	12.91 %	771 k$
H3, CT, UR	1.20 %	1,989 k$	1.64 %	12,280 k$	9.79 %	1,666 k$	12.82 %	688 k$
H3, CT, SRI, UR	1.20 %	2,014 k$	8.34 %	10,889 k$	10.04 %	1,499 k$	13.14 %	798 k$

7.8.2 Cybercrime Group

In this threat scenario, shown in the leftmost column of Table 7.5, we consider a hacking group that can compromise NS resolutions and exploit XSS vulnerabilities, the most widespread type of vulnerabilities in web applications according to the OWASP foundation (see Figure 7.5 for the Pareto frontier).We look at the MyEtherWallet attack from 24th of April 2018 [tea18]. The attack started by hijacking Amazon's *Route 53* name servers via BGP. The attackers rerouted requests to this name server to a malicious server that referred the users to a phishing website imitating MyEtherWallet. While our model excludes BGP hijacking as an attack vector due to the possible global exposure, we instead model the situation where the hacker group compromised the name servers directly. With 2% of all page views on the Alexa top 5000, the impact is already considerable. The attacker compromises the DNS resolution for a set of CDNs, like cdn-1.tstatic.net and content.jwplatform.com. This approach allows to compromise all the websites that rely on these CDNs for the inclusion of JS resources. Furthermore, the Amazon *Route 53* DNS servers are queried for the DNS resolution of many domains, such as reddit.com, twitter.com or dropbox.com. IPsec, DNSSEC and DANE have no effect, because we assumed the DNS servers themselves to be compromised.[22] The most effective countermeasure is to employ secure connections both for the website

[22] For the actual BGP-based attack, the attacker cannot sign in the name server's stead, hence DNSSEC/DANE could improve the situation.

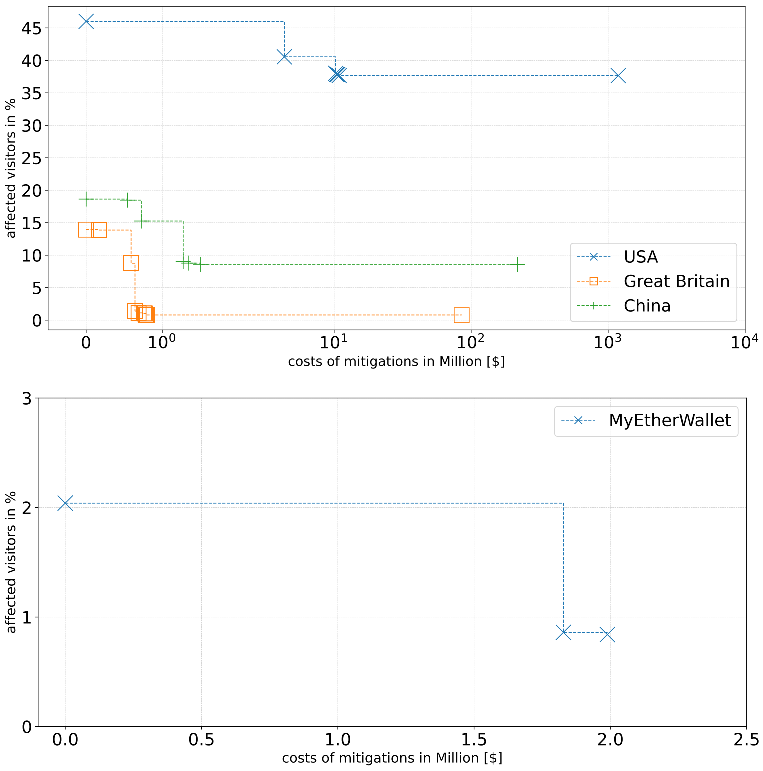


Figure 7.5: Pareto frontiers for state-controlled and small hacker group attacker scenarios.

via HTTPS, HTTPS-Redirect and HSTS (abbreviated H3 in the following) and the external JS inclusions. Combining H3 with upgrade-insecure-requests (abbreviated UR in the following), we achieve the maximum increase in security. This matches the application-level countermeasures proposed by the developers in the aftermath [tea18]. In summary, enforcing endpoint level defense is the optimal solution for this threat.

7.8.3 Nation-State Groups

In this scenario (Table 7.5, right-side columns), we consider the potential of three states to mount an attack, assuming that local legislation permits such an attack (see Figure 7.5 for the Pareto frontier for each country). The Great Cannon attack, e.g., is believed to have been mounted from China.

The US is the country with the highest attack potential: about 46% of the visits on the Top 5K are affected. By applying different mitigations, this reward can only be reduced to 38%. Due to the importance of domains under US jurisdiction, many page views would be directly compromised. The potential impact of China and GB is much smaller. Where GB's attack potential can be reduced from about 14% to about 1%, China's attack potential can only be reduced from 19% to 9%, which can be explained by the relative autonomy of the Chinese Internet infrastructure. However, the single most effective mitigation is IPsec, being nearly as effective as the combination of H3, CT, SRI and, with marginal impact, UR. These mitigations are protecting foreign websites that rely on Chinese infrastructure for routing, resolution or content distribution, but not Chinese websites. GB has influence on foreign pages as well, but a larger share of them are able to deploy helpful countermeasures. In particular, GB is the best scenario to demonstrate the viability of IPsec, single-handedly reducing the attacker success from 14% to 2%. This is likely because of the GB's access to transatlantic submarine cables. By contrast, infrastructure that is routed via the US and China is often situated in the same country, due to their size relative to their neighbors and China's stated goal of self-reliance.

From Table 7.5, SRI and then H3+SRI are the cheapest mitigation for the US, it is H3 and then SRI for China and GB. In all three cases, the optimal countermeasure is SRI, UR, H3, CT, and IPsec (USA, CN) or SRI, H3, and IPsec (GB).

7.8.4 DANE vs. Certificate Transparency

To evaluate DANE, which proactively mitigates certificate forgeries, we considered a scenario where we artificially removed CT. This has the same effect as forgoing the sneakiness assumption concerning after-the-fact detection of certificate forgeries.

Note first that DNSSEC is a prerequisite to DANE and recall that it is not applicable on all hosts. We find that the improvement of deploying DANE (asserting the current end-entity certificate) in addition to DNSSEC is zero in all scenarios. The reason is as follows: DANE is only effective if, in addition to the domain and its NSs, all CDNs that provide JS inclusions deploy DNSSEC. The majority of JS providers do not. We inspected the remaining cases and, while DANE thwarts attacks based on certificate compromise, other attacks (mostly on JS inclusions) still apply. Even applying H3 where possible, the improvement from adding DANE remains zero.

7.8.5 Discussion

Overall, we find that the influence of the biggest players on the market, in particular Google, is significant and comes close to the adversarial capabilities of a state-sponsored attacker. At the same time, we find that regardless of the type of attacker we consider, securing the most popular domains can be primarily achieved by deploying endpoint mitigations such as HTTPS, HSTS, and SRI. This is sometimes augmented by the use of UR. Additionally, IPsec plays a significant role at securing against the countries but not against the service providers.

Moreover, deploying these comparatively cheap endpoint mitigations allows to quarter the user's exposure against infrastructure attackers with a cost of less than $6 M. On average, this amounts to about $1,000 per domain. The exception is the Google scenario, where such a decrease is not possible. Likewise, there is little defense against the US.

Despite the sneakiness assumption, DNSSEC achieves little at high cost. Even though, theoretically, amortized cost could make these countermeasures a viable alternative considering the number of domains, this is not the case. On the other hand, our analysis has indicated that IPSec is an effective, although extraordinary expensive, mitigation against China and GB.

7.8.6 Performance

Table 7.6: Performances for Alexa Top 5000. The property graph used in these scenarios has 70,975 nodes and 329,899 edges.

	attack graph generation			status quo analysis (s)	applying mitigation (s)			current efficacy (s)		
scenario	runtime(s)	# nodes	# edges		min	med	max	min	med	max
US	9843.02	129,371	2,191,112	49.05	59.58	108.98	284.75	94.27	405.65	585.78
CN	558.57	43,812	252,343	6.36	9.87	24.85	35.66	10.91	52.13	83.06
GB	215.16	28,626	50,948	1.55	3.77	14.40	23.71	1.81	60.98	92.45
MyEtherWallet	48.36	12,568	26,010	0.28	1.10	9.96	19.27	0.24	166.70	243.98
Google	125.16	31,598	73,148	1.75	5.15	15.75	24.61	3.08	91.05	143.70
Amazon	979.73	62,131	167,317	8.45	14.16	24.33	34.83	11.60	76.74	115.64
Godaddy	111.44	27,652	33,770	0.89	2.92	13.65	22.20	1.30	40.88	75.76
CloudFlare	345.74	18,293	35,215	0.74	2.09	10.55	21.41	0.70	94.17	143.30
Dyn	33.15	5,152	5,387	0.1	1.43	9.79	18.72	0.08	77.85	116.25

The graph-based analysis algorithm discussed in Section 7.7 reduces the analysis effort for a given mitigation by precomputing the attack graph from the property graph. The runtime of this precomputation step (called 'attack graph generation' in Table 7.6) depends on the scenario of choice, ranging from about one minute for Dyn to 3 h for the US, the country with the largest attack graph. The size of the generated attack graph governs the time the status quo analysis takes, as it is a simple reachability query. Neo4j is optimized for such queries, hence, even for the US attack graph that contains about two million edges, the analysis takes less than a minute. This makes it feasible to analyze different mitigation scenarios (which remove edges from the attack graph) and analyze the efficacy of existing mitigations (which adds edges to the attack graph). We combine the time to remove or add edges with the runtime of the reachability query and report the minimum, median and maximum. As expected, there is quite a range: queries that modify the graph are more expensive than reachability queries. Hence, the more edges a mitigation removes, the higher the runtime. Half the queries in the largest attack graph take less than two minutes. Computing the data needed for mitigation-web.github.io, i.e., generating the attack graphs and computing the potential and efficacy for all 256 combinations of 8 mitigations, took about 52 h in total. All computations were performed on an Intel Xeon E5-4650L @ 2.60GHz. Because a Neo4j Cypher query is always computed in a single thread, we only made use of one CPU core. Further, 32 Gb RAM was sufficient.

7.9 Related Work

The vulnerability of the Internet at the infrastructure level has been studied before [Gol14; BFM+10], including the European BGP topology [FER+16], and web attacks [SPR+17b], but the analysis of mitigations has been largely ignored.

7.10 Conclusion

We proposed a holistic approach for securing the users from web-based attacks, based on a comprehensive Stackelberg planning model of attacks and defenses (with associated costs and security benefits), and an optimized graph-based algorithm. We analyzed the susceptibility of the top 5K Alexa domains against attackers ranging from cyber-criminal groups to infrastructure providers and nation-state actors. We find that large infrastructure providers are almost as powerful as nation-state attackers. We were able to compute solutions that significantly increase the security of the users. Interestingly, while significant effort has been spent to develop and deploy high-cost mitigations like IPsec or DNSSEC, our analysis highlights that the increase in security is enabled merely by the usage of cheap endpoint defenses like HTTPS, HSTS, and SRI.

Our approach is easy to extend and adapt, and thus provides a foundation for future analyses at web scale. For example, it can be easily extended with additional mitigations like CSP. Another potential target is non-physical dependencies, for instance, when a domain's TLS implementation shares the Diffie-Hellman group with others and is thus susceptible to attacks with reasonable cost-per-domain [ABD+15]. Likewise, new technological proposals to improve web security can immediately be added to the mitigation model to compete against existing technologies.

Part IV

Prototypes

Stackelberg planning based tools made easily accessible via web-based user interfaces

8

Planning in the Browser

8.1 Motivation and Contributions

'Move fast and break things often' — Facebook's developer motto until 2014 — summarizes the mentality of modern web development: web application development is an agile process with frequent specification changes. At the same time, there is a shift to push computation to the client side, i.e., the browser, motivated by both economic and operational reasons. User-side computation is scalable by design, averts security and availability issues that come with the allotment of server-side resources and saves cost (by externalizing them to the user). But there are also operational reasons: web applications can be 'closer' to the user, their personal habits and preferences. We can solve privacy-related challenges by simply not running the code in the cloud, while retaining the mobility of cloud-based code delivery.

Planning technology is a perfect match for 'move fast'-mentality: solvers are drop-in solutions; they are not necessarily performance-optimal, but fast. They can quickly adapt to frequent problem changes. Unfortunately, no competitive planner is available in the browser, where JavaScript is the lingua franca for active content. Moreover, traditional planners are typically heavily optimized C or C++ code. Even if rewritten in JavaScript, their performance would suffer, e.g., due to JavaScript's dynamic type checks, its garbage collector or lack of ahead-of-time compiler optimization.

WebAssembly (WASM), which has become a W3C recommendation in 2019 (W3C), provides a solution to both problems. It defines a portable binary-code format for executable programs and was designed to run at near-native speed. With Emscripten[1], there is a compiler from C to WASM that helps porting existing planners. Considering the superior performance of WASM over JavaScript, a generic planning algorithm in WASM may end up faster than a specialized algorithm implemented in JavaScript.

[1] https://emscripten.org/

We used Emscripten to port the Fast Downward Planning System [Hel06], including the extensions to Stackelberg planning from Part I. We will skim over some challenges encountered in porting before, but focus on existing and potential applications for in-browser planner. Finally, we evaluate the performance in WASM versus a native execution and show that the overhead is affordable (50–100%).

8.2 Porting Challenges

Emscripten is a source-to-source compiler that runs as a back end to the LLVM compiler and produces a subset of JavaScript known as WASM. The challenges were threefold.

- Fast Downward consists of two modules, the translator which reads an input PDDL file and transforms it to an intermediate FDR representation, and the search component which solves the task. Porting the translator part, which is written in Python, to WASM is not straightforward. In the demo, we substitute the translator by a domain-specific encoding of our domain in FDR.
- Portability: Fast Downward contains OS-specific code for memory usage statistics and input handling. As the browser's execution environment does not provide them, they had to be replaced or removed.
- Transmitting input/output: Fast Downward employs C++-style input/output streams which are incompatible with the browser's execution environment. By default, Emscripten writes the output as a file within a virtual file system. We modified Fast Downward to obtain the input file as a command line string, and added glue code that fetches the output file and displays its content.

8.3 Demonstration

Our demonstration consists of two parts, a demonstration for the case study conducted in Section 6.7 that runs the planner in the browser, and a simple interface that allows the user to run the planner on any instance provided by the user. Both are available under project.cispa.io/fd-in-browser.

8.3.1 Analyze Case Study From Section 6.7 With Customizable Parameters

In Chapter 6, we used Stackelberg planning to compare various security technologies (e.g., IPsec, DNSSEC, SMTP-over-TLS, STARTTLS, strong certificate validation in SMTP) to improve confidentiality in the email system. Recapping, the first player deploys one or more of these technologies on a set of hosts in the current infrastructure. The second player represents a surveillance attacker and spies on as many connections as possible. The outcome is a front of plans that are Pareto-optimal w.r.t. the cost of deploying the countermeasures, and the users affected by the attack.

A major obstacle in communicating these results is the agreement on cost estimates, which vary wildly between countries and companies operating the infrastructure. Yet a result is only meaningful to a user if they consider these parameters accurate. We

Table 8.1: Performance evaluation for Native (N), Chrome (C), and Firefox (F). The average search time is computed over all covered tasks. On the remaining 3 tasks per domain hat take more than 0.3s on all platforms, we computed the overhead as the arithmetic mean of the native runtime divided by the runtime in the respective browser minus 1, here presented in % ± standard deviation (std).

	logistics98	rovers	satellite
coverage (N/C/F)	6/6/6 of 35	7/7/7 of 40	7/7/7 of 36
coverage $> 0.3s$ (N/C/F)	3/3/3 of 35	3/3/3 of 40	3/3/3 of 36
avg. time Native	8.95s	1.98s	10.00s
avg. time Chrome	16.23s	3.15s	15.85s
avg. time Firefox	17.48s	3.41s	16.40s
overh. Chrome	83.2 ± 13.4	58.0 ± 6.38	72.2 ± 15.3
overh. Firefox	93.1 ± 11.2	72.2 ± 5.11	87.0 ± 20.6

provide a ‘parametric’ version of our results as a web application (see Figure 8.1) that allows for customizing these parameters. Users can now refine our estimates according to their expertise and, possibly, insider knowledge, without having to share it with other partners. They can also choose among several attacker models and groups of users to be protected, providing highly individualized results, if desired.

8.3.2 Planning in the Browser

We offer a simple interface shown in Figure 8.2, where users can provide a task in FDR representation and run Fast Downward to solve it. This allows to evaluate the performance of the in-browser planner before implementing a domain-specific user interface. The user can choose between an optimal (A*with $h^{\text{LM-cut}}$ [HD09]) and an agile (LAMA [RWH11]) configuration.

8.4 Evaluation

We measured the performance impact of running Fast Downward inside the browser environment. We run A^* search with the LM-cut [HD09] heuristic on three standard IPC domains, namely Logistics98, Rovers, and Satellite. We ran the experiments on an Intel Core i7-7820HQ, 2.90 GHz with Chrome v85 and Firefox v80.0. For each task, we set a 5-minute time limit and a 4-GB memory limit. All tasks solved with the native implementation also finished in the browsers, and the memory limit was never hit.

For the calculation of the overhead metric, we ignored all tasks taking less than 0.3s on all platforms, because it would be biased by the *near zero numbers* otherwise. Thus, across all the tasks finishing in less than 5 minutes, but more than 0.3s, there is a performance loss of around 50% to 100% (see Table 8.1). This number is in line to the previous findings reporting an average slowdown of 45% (Firefox) to 55% (Chrome), with peak slowdowns of 2.08× (Firefox) and 2.5× (Chrome) [JPB$^+$19]. Our results are slightly better, possibly because Fast Downward does not need to perform slow IO operations after reading in the input file.

8.5 Conclusion and Potential Applications

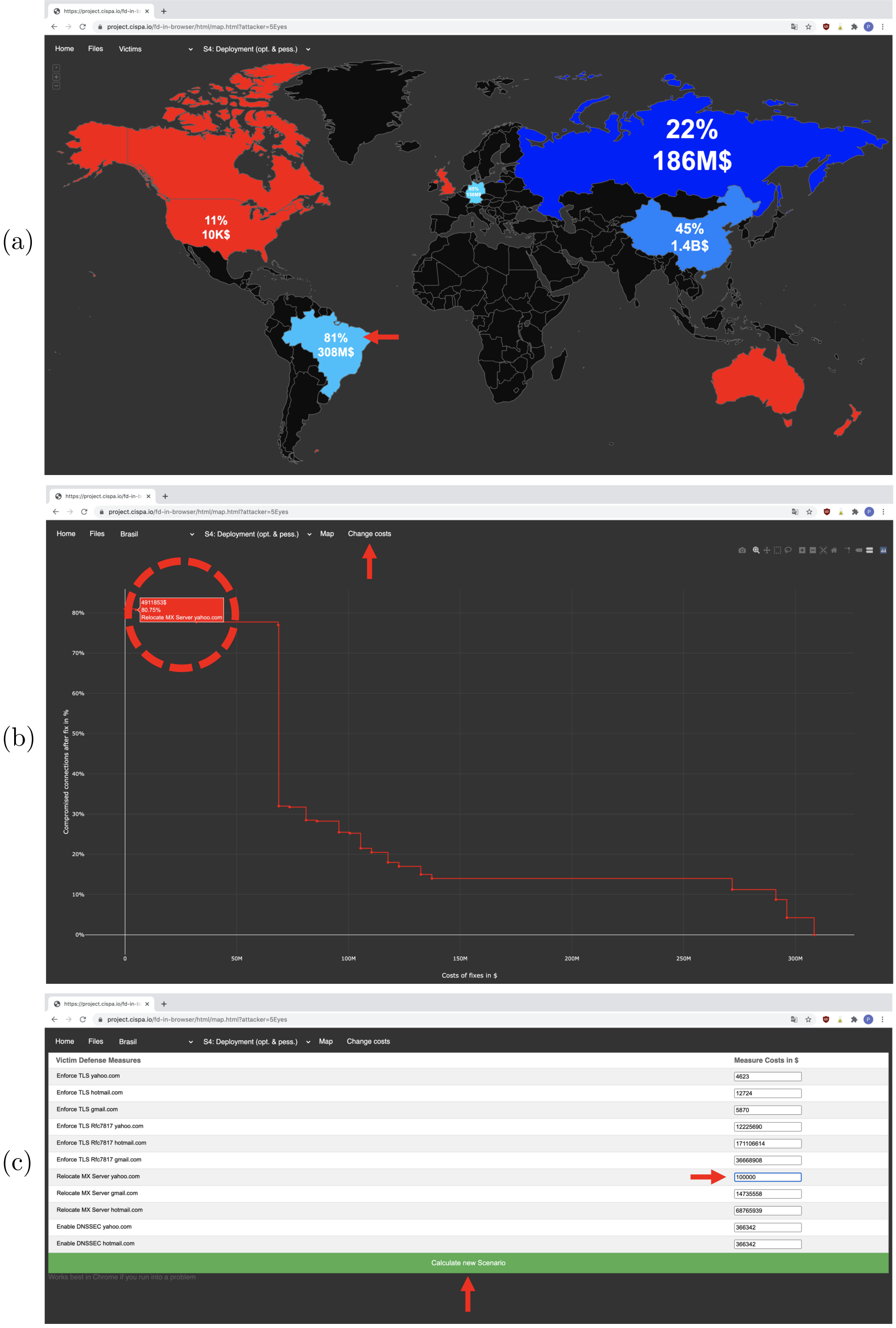
This demo showcases how planners can be run in the browser to improve scalability and privacy compared to computation in the cloud, and avoid having to compile and install the planner, compared to native execution. Thus we foresee many potential applications.

8.5.1 Ready-To-Go Planning IDE

An in-browser planner provides the possibility to edit and solve PDDL input directly in the browser. This facilitates the quick and easy use of planning technology. Existing in-browser editors like https://editor.planning.domains and WEB PLANNER [MPM+17] provide this functionality, but rely on a backend server for problem solving. Besides the aforementioned issues in terms of scalability, availability and privacy of the input data, the network latency adds to the perceived responsiveness, which is important in small domains, as used in teaching. Integrating our WASM module would immediately mitigate all these issues: results would display as soon as they are computed and users could exploit the full power of their machine.

8.5.2 Data Privacy Plugins

Data privacy research suggests the sanitization of data sent to social networks to minimize the risk of exposure while retaining the utility of the information. Examples are the perturbation of location information [LAT+20], the generalization of hashtags in tweets [ZHR+18] or the redaction of images [OFS18]). Planning algorithms can be used to optimize these steps if the sanitization can be described in terms of discrete operations [KHS+18]. Sanitization would typically be implemented in browser plug-ins, as these can interact with the social network website while also being under the user's control.

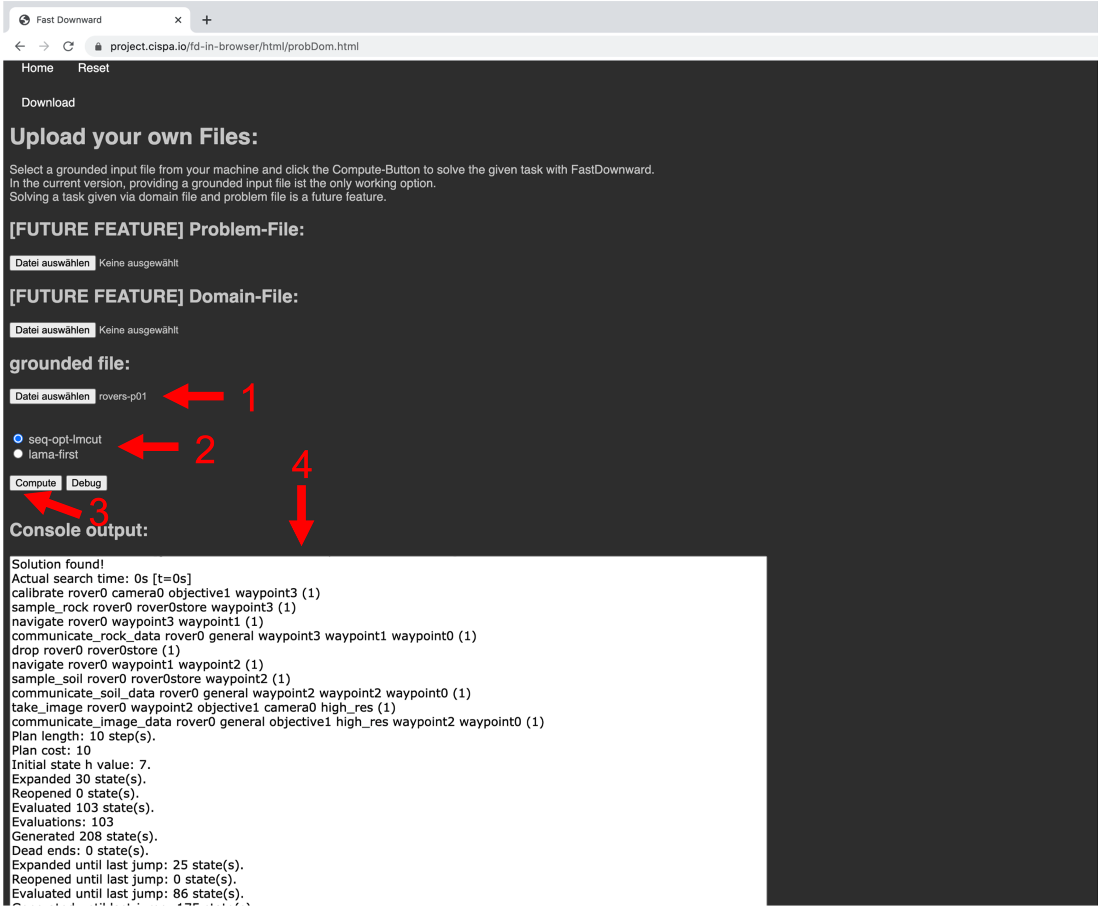

(a)
https://project.cispa.io/fd-in-b
project.cispa.io/fd-in-browser/html/map.html?attacker=5Eyes
Home
Files
Victims
S4: Deployment (opt. & pess.)
22%
186M$
11%
10K$
45%
1.4B$
81%
308M$
(b)
Home
Files
Brasil
S4: Deployment (opt. & pess.)
Map
Change costs
4911853$
80.75%
Relocate MX Server yahoo.com
80%
70%
60%
50%
40%
30%
20%
10%
0%
Compromised connections after fix in %
0
50M
100M
150M
200M
250M
300M
Costs of fixes in $
(c)
Home
Files
Brasil
S4: Deployment (opt. & pess.)
Map
Change costs

Victim Defense Measures	Measure Costs in $
Enforce TLS yahoo.com	4623
Enforce TLS hotmail.com	12724
Enforce TLS gmail.com	5870
Enforce TLS Rfc7817 yahoo.com	12225690
Enforce TLS Rfc7817 hotmail.com	171106614
Enforce TLS Rfc7817 gmail.com	36668908
Relocate MX Server yahoo.com	100000
Relocate MX Server gmail.com	14735558
Relocate MX Server hotmail.com	68765939
Enable DNSSEC yahoo.com	366342
Enable DNSSEC hotmail.com	366342

Calculate new Scenario

Works best in Chrome if you run into a problem

Figure 8.1: Screenshot of GUI for Chapter 6 case study. (a) map of defender & attacker countries and attacker reward & defender cost; (b) Pareto frontier of mitigations + additional info; (c) freely edit the costs of all mitigations and by clicking on the “Calculate new Scenario”-Button, Fast Downward computes the Pareto frontier with your costs directly in the browser



Upload your own Files:

Select a grounded input file from your machine and click the Compute-Button to solve the given task with FastDownward.
In the current version, providing a grounded input file ist the only working option.
Solving a task given via domain file and problem file is a future feature.

[FUTURE FEATURE] Problem-File: Keine ausgewählt

[FUTURE FEATURE] Domain-File: Keine ausgewählt

grounded file: rovers-p01

● seq-opt-lmcut
○ lama-first

Console output:

```
Solution found!
Actual search time: 0s [t=0s]
calibrate rover0 camera0 objective1 waypoint3 (1)
sample_rock rover0 rover0store waypoint3 (1)
navigate rover0 waypoint3 waypoint1 (1)
communicate_rock_data rover0 general waypoint3 waypoint1 waypoint0 (1)
drop rover0 rover0store (1)
navigate rover0 waypoint1 waypoint2 (1)
sample_soil rover0 rover0store waypoint2 (1)
communicate_soil_data rover0 general waypoint2 waypoint2 waypoint0 (1)
take_image rover0 waypoint2 objective1 camera0 high_res (1)
communicate_image_data rover0 general objective1 high_res waypoint2 waypoint0 (1)
Plan length: 10 step(s).
Plan cost: 10
Initial state h value: 7.
Expanded 30 state(s).
Reopened 0 state(s).
Evaluated 103 state(s).
Evaluations: 103
Generated 208 state(s).
Dead ends: 0 state(s).
Expanded until last jump: 25 state(s).
Reopened until last jump: 0 state(s).
Evaluated until last jump: 86 state(s).
Generated until last jump: 175 state(s).
```

Figure 8.2: Screenshot for Planning in the browser GUI. 1) select an arbitrary grounded FDR planning task on your machine; 2)choose between two configurations of Fast Downward, namely seq-opt-lmcut and lama-first; 3) click on "Compute"-button and Fast Downward solves your planning task directly in your browser using WebASM code; 4) console output is shown in a text box

9 Publicly Available Implementations

For this thesis, we developed several research prototypes to evaluate our algorithmic innovations and to test our approaches. We also designed web-based graphical user interfaces to not only visualize our case studies but also allow the interested reader to interact and play around with it. In particular, the following tools are available online:

(1) The Stackelberg planning implementation from Chapter 3 as an extension of the Fast Downward framework [SST21].

(2) The Symbolic Stackelberg planning source code, benchmarks, and results from Chapter 4 as a derivative of our initial Stackelberg planning implementation [TSK$^+$21a].

(3) The specialized algorithms for Chapter 6 built in the Fast Downward framework as well [SST21].

(4) The website (project.cispa.io/fd-in-browser) from Chapter 8 to interact with the case study from Chapter 6 and solve arbitrary Planning tasks in the browser [STT20].

(5) The fork of a recent version of Fast Downward which makes the necessary changes to enable compilation of the search and the translator to WebAssembly and thus allows everyone to run Fast Downward in their browser [TS20].

(6) The website (mitigation-web.github.io) from Chapter 7 to visualize and allow interaction with the case study conducted there [Spe22].

The implementations for Chapter 5 and Chapter 7 are currently not publicly available, but we plan to change that soon.

10
Conclusion

10.1 Summary of Contributions

This dissertation presented a line of work that holistically tackles simulated penetration testing and mitigation analysis. In particular, we made the following threefold contributions: 1) formal and algorithmic groundwork in Part I, 2) applying the formalism and algorithms to local network and Internet-wide applications in Part II and Part III, and 3) making the Fast Downward planner including Stackelberg extensions easily accessible for everyone by enabling it to run in a web browser and demonstrating it with a case study in Part IV. We summarize the contributions of the chapters in this dissertation in the following.

We established Stackelberg planning in Chapter 3 and laid the formal and algorithmic foundations for the remainder of the thesis. Stackelberg planning is an exciting new variant of planning that has proven to be of high practical value in a range of potential applications. It allows modeling defenses against attackers encoded as planning agents and it subsumes a notion of robustness against sabotage and against worst-case perturbances by an environment. From an algorithmic point of view, it provides an interesting middle ground between full-scale game-theoretic planning and classical planning, generalizing the latter to a single adversarial exchange of action sequences.

We introduced Symbolic Leader Search (SLS) in Chapter 4, an advanced algorithm for solving Stackelberg planning tasks and, at the same time, showed that successful algorithmic techniques from classical planning can be lifted to the Stackelberg planning setting. SLS exploits symbolic leader search and cost-bounded follower search to share information between subtasks. It thus consistently outperforms our previous approach, in particular when the leader action space is large or when we consider soft goals.

We proposed the first comprehensive theory for privilege escalation and demonstrated its use in improving formal threat models in Chapter 5. It exposes a hierarchy of hierarchies that can be used to categorize vulnerabilities and instantiate the effect of exploits precisely in many cases. We demonstrated that it is both cheap and useful to augment the well-established CVSS standard by an additional field and obtain a clearly defined semantics for this field. Our own security scanner and risk analysis tool, CATSS/MAUS, provides the first non-commercial tool to perform on-system scans and thus produce detailed network models. Its modular structure provides a basis for new risk estimation techniques. Capitalizing on our Stackelberg planning foundations, we implemented our theory in MAUS, demonstrated the quality of our formal model, and measured the runtimes of our framework on practical examples.

With the aid of a Stackelberg Planning model and domain-specialized solver, we showed that a holistic analysis of deployment benefits of cryptographic protocols, secure configurations and political measures is possible with a high degree of automation in Chapter 6. Based on a very simple cost assessment in the case of email communication, one finding amongst others is that the enforcement of TLS would have a great effect in most countries. If strict TLS validation can be achieved, RFC 7817 should be implemented with it: despite its simplicity, it reduces mitigation cost by more than 20%. However, it is plausible that the hidden cost of deployment, a loss of functionality, make this approach impractical. Further,while the US, Russia and China are relatively self-sufficient, the privacy of email users in Brazil is highly vulnerable due to foreign dependencies, requiring costly relocation.

In Chapter 7, we proposed a holistic approach for securing the users from web-based attacks, based on a comprehensive Stackelberg planning model and a novel graph-based algorithm. We analyzed the susceptibility of the top 5K Alexa domains against attackers ranging from cyber-criminal groups to infrastructure providers and nation-state actors. We find that large infrastructure providers are almost as powerful as nation-state attackers. We were able to compute solutions that significantly increase the security of the users. Interestingly, while significant effort has been spent to develop and deploy high-cost mitigations like IPsec or DNSSEC, our analysis highlights that the increase in security is enabled merely by the usage of cheap endpoint defenses like HTTPS, HSTS, and SRI. Our approach is easy to extend and adapt, and thus provides a foundation for future analyses at web scale. For example, it can be easily extended with additional mitigations like CSP.

Finally, we illustrated how we ported the Fast Downward Planning System, including the extensions to Stackelberg planning, s.t. it can be compiled to WASM and run in a browser in Chapter 8. We evaluated the performance in WASM versus a native execution and showed that the overhead is affordable (50–100%). With a case study for Chapter 6, we showcased how Fast Downward can be run in the browser to improve scalability and privacy compared to computation in the cloud, and avoid having to compile and install the planner, compared to native execution.

10.2 Future Research Directions

The author of this dissertation envisions different future research directions that continue all three areas of this work, namely model and algorithms, applications, and prototypes.

We have already shown that successful algorithmic techniques from classical planning can be lifted to the Stackelberg planning setting, e.g., symbolic search. Thus, the first major area for future work is to identify and implement other promising techniques. To be more specific, the fast computation of the Pareto frontier depends on the speed with which a first good solution, a cheap fix-action sequence reducing attacker success probability to a small value, can be found. In case that happens quickly, our pruning methods and thus the search become highly effective; in case it does not happen quickly, the search often becomes prohibitively enumerative and can take too much time. A classical planning search is typically guided by a heuristic function and we also do that in the follower search of our current Stackelberg planning solvers, but not for the leader search. Therefore, future works could, amongst others, explore the possibilities of guiding the search also at leader level to find good solutions earlier.

The second future work area is more application scenarios. We already showed the applicability of our approach to a local network and at Internet scale. The different applications vary in complexity, e.g., we explained how a simple penetration testing model can be refined with a notion of privilege escalation in Chapter 5. There is always a trade-off between the accuracy of the model and the practicality of generating and solving the model. If a model is more refined, it is, of course, a better reflection of the real world. On the flip side, more detailed models are harder to create automatically and can require more manual effort. In addition, it can take much more time to compute a solution for a detailed model. Future works could tackle acquiring more data about the network and systems under investigation or could explore other interesting scenarios, e.g., DDoS attacks and corresponding defenses, while at the same time ensuring that the models are still feasible regarding computational complexity and required manual effort.

Lastly, the third future work area is making simulated penetration testing and mitigation analysis and planning more accessible for, e.g., the management level of corporations, politicians and a general audience. By porting Fast Downward to WASM, we already paved way for more web-based applications. One example is a ready-to-go planning IDE (see Section 8.5) like https://editor.planning.domains which currently relies on a backend server for problem solving. Integrating our WASM module would allow users to exploit the full power of their machine without availability, privacy issues, and network latency of the backend server. As another example, we implemented a prototype web-based GUI which can display the topology of the local computer network under investigation, the found vulnerabilities, and the Pareto frontier of suggested mitigations in Chapter 5. We envision an even more advanced and easy-to-understand web-based dashboard from which one can initiate and investigate mitigation analyses with only a few clicks.

Bibliography

Author's Papers for This Thesis

[PSSB+18] Speicher, P., Steinmetz, M., Backes, M., Hoffmann, J., and Künnemann, R. Stackelberg planning: towards effective leader-follower state space search. In McIlraith, S. and Weinberger, K., editors, *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)*, 6286–6293. AAAI Press, February 2018.

[PTSK+21a] Torralba, Á., Speicher, P., Künnemann, R., Steinmetz, M., and Hoffmann, J. Faster stackelberg planning via symbolic search and information sharing. In Leyton-Brown, K. and Mausam, editors, *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI'21)*. AAAI Press, January 2021.

[PSBK+22] Speicher, P., Battiston, G., Künnemann, R., and Backes, M. Playing catss and maus: a theory of privilege escalations and applications to simulated pentesting. In *not published yet*, 2022. forthcoming.

[PSSK+18a] Speicher, P., Steinmetz, M., Künnemann, R., Simeonovski, M., Pellegrino, G., Hoffmann, J., and Backes, M. Formally reasoning about the cost and efficacy of securing the email infrastructure. In *Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P'18)*, 77–91, 2018.

[PTSS+22] Tizio, G. D., Speicher, P., Simonovsky, M., Backes, M., Stock, B., and Künnemann, R. Pareto-optimal defenses for the web infrastructure: theory and practice. *ACM Transactions on Privacy and Security (TOPS)*, October 2022.

[PTSK+20] Tran, N., Speicher, P., Künnemann, R., Backes, M., Torralba, A., and Hoffmann, J. Planning in the browser. In *System Demonstration at the 30th International Conference on Automated Planning and Scheduling (ICAPS'20)*, 2020.

Other Papers of the Author

[SFGS+18] Fickert, M., Gnad, D., Speicher, P., and Hoffmann, J. Saarplan: combining saarland's greatest planning techniques. In *IPC 2018 planner abstracts*, 2018.

[SSSH+19] Speicher, P., Steinmetz, M., Hoffmann, J., Backes, M., and Künnemann, R. Towards automated network mitigation analysis. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 1971–1978, 2019.

Publicly Available Implementations for This Thesis

[TSpe22] Speicher, P. Web-based interactive visualization of the case study for pareto-optimal defensive strategies for securing the web, 2022. URL: https://mitigation-web.github.io/.

[TSST21] Speicher, P., Steinmetz, M., and Torralba, A. An extension of fast downward that implements the stackelberg planning algorithms, 2016-2021. URL: https://bitbucket.org/PatrickSpeicher/sim-pentest-what-if/.

[TSTT20] Speicher, P., Tran, N., and Torralba, A. Planning in the browser by compiling fast downward to webassembly, 2020. URL: https://project.cispa.io/fd-in-browser/.

[TTSK+21a] Torralba, Á., Speicher, P., Künneman, Steinmetz, M., and Hoffmann, J. Code, benchmarks, and data of faster stackelberg planning via symbolic search and information sharing. https://doi.org/10.5281/zenodo.4320574, 2021.

[TTS20] Tran, N. and Speicher, P. Fork of fast downward which enables compilation to webassembly, 2020. URL: https://github.com/abwesend890/downward-in-the-browser.

Other References

[ISC20] (ISC)^2. CYBERSECURITY WORKFORCE STUDY. Technical report, 2020.

[tea18] #teamMEW, M. A message to our community - a response to the DNS HACK of april 24th 2018, 2018. URL: https://medium.com/@myetherwallet/a-message-to-our-community-a-response-to-the-dns-hack-of-april-24th-2018-26cfe491d31c.

[AL06] Abley, J. and Lindqvist, K. Operation of Anycast Services. RFC 4786, IETF, 2006.

[Ade16] Adestra. Primary e-mail providers according to consumers in the united states as of 2016, 2016.

[Adk14a] Adkins, M. Massive growth in smtp starttls deployment, 2014.

[Adk14b] Adkins, M. The current state of smtp starttls deployment, 2014.

[ABD+15] Adrian, D., Bhargavan, K., Durumeric, Z., Gaudry, P., Green, M., Halderman, J. A., Heninger, N., Springall, D., Thomé, E., Valenta, L., VanderSloot, B., Wustrow, E., Zanella-Béguelin, S., and Zimmermann, P. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In, 2015.

[Afi21] Afifi-Sabet, K. Google is shifting youtube infrastructure to google cloud, June 2021. https://www.itpro.co.uk/cloud/cloud-computing/359785/google-is-shifting-youtube-infrastructure-to-google-cloud.

[AWK02] Ammann, P., Wijesekera, D., and Kaushik, S. Scalable, graph-based network vulnerability analysis. In *ACM Conference on Computer and Communications Security*, 217–224, 2002.

[Ano14] Anonymous. Towards a comprehensive picture of the great firewall's DNS censorship. In, 2014.

[APN] APNIC. Dnssec validation rate by country. Accessed: 01/09/2021 URL: https://stats.labs.apnic.net/dnssec.

[ACF$^+$20] Aqeel, W., Chandrasekaran, B., Feldmann, A., and Maggs, B. M. On landing and internal web pages: the strange case of jekyll and hyde in web performance measurement. In, 2020.

[AAL$^+$05a] Arends, R., Austein, R., Larson, M., Massey, D., and Rose, S. DNS Security Introduction and Requirements. RFC 4033, IETF, 2005.

[AAL$^+$05b] Arends, R., Austein, R., Larson, M., Massey, D., and Rose, S. Protocol Modifications for the DNS Security Extensions. RFC 4035, IETF, 2005.

[BN95] Bäckström, C. and Nebel, B. Complexity results for SAS$^+$ planning. *Computational Intelligence*, 11(4):625–655, 1995.

[BHH11] Baker, J., Hansbury, M., and Haynes, D. The oval language specification. *MITRE, Bedford, Massachusetts (url: http://ebookbrowsee.net/ovallanguage-specification-08-08-2011-pdf-d222972411)*, 2011.

[BS15] Barth, A. and Sterne, B. Content Security Policy 1.0. Technical report, W3C, 2015. URL: http://www.w3.org/TR/2015/NOTE-CSP1-20150219/.

[BHR$^+$12] Bayuk, J. L., Healey, J., Rohmeyer, P., Sachs, M. H., Schmidt, J., and Weiss, J. *Cyber security policy guidebook.* John Wiley & Sons, 2012.

[BD12] Bilge, L. and Dumitraş, T. Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, 833–844, Raleigh, North Carolina, USA. Association for Computing Machinery, October 16, 2012.

[Blo21a] Blog, C. A safer default for navigation: https, 2021. URL: https://blog.chromium.org/2021/03/a-safer-default-for-navigation-https.html.

[Blo21b] Blog, M. S. Firefox 91 introduces https by default in private browsing, 2021.

[Blu17] Blum-Dumontet, E. Who's That Knocking At My Door? Understanding Surveillance In Thailand. Technical report, Privacy International, 2017.

[BGH$^+$05] Boddy, M., Gohde, J., Haigh, T., and Harp, S. Course of action generation for cyber security using classical planning. In Biundo, S., Myers, K., and Rajan, K., editors, *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05)*, 12–21, Monterey, CA, USA. AAAI Press, 2005.

[BMS$^+$12] Bonet, B., McCluskey, L., Silva, J. R., and Williams, B., editors. *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, 2012. AAAI Press.

[Bor16] Borders, R. W. World press freedom index 2016, 2016. Retrieved 03 May 2017.

[BFK$^+$13] Borrajo, D., Fratini, S., Kambhampati, S., and Oddi, A., editors. *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13).* Rome, Italy, 2013. AAAI Press.

[Bou09] Boutilier, C., editor. *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09).* Pasadena, California, USA, July 2009.

[BJV03] Bowling, M. H., Jensen, R. M., and Veloso, M. M. A formalization of equilibria for multiagent planning. In Gottlob, G., editor, *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, 1460–1462, Acapulco, Mexico, August 2003.

[BDE$^+$09] Brafman, R. I., Domshlak, C., Engel, Y., and Tennenholtz, M. Planning games. In [Bou09], 73–78.

[Bry86] Bryant, R. E. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[BCF$^+$12] Bugliesi, M., Calzavara, S., Focardi, R., and Squarcina, M. Gran: Model Checking Grsecurity RBAC Policies. In *2012 IEEE 25th Computer Security Foundations Symposium.* 2012 IEEE 25th Computer Security Foundations Symposium (CSF), 126–138, Cambridge, MA, USA. IEEE, June 2012.

[BFM$^+$10] Butler, K. R. B., Farley, T. R., McDaniel, P. D., and Rexford, J. A survey of BGP security issues and solutions. *Proceedings of the IEEE*, 98, 2010.

[CRB16] Calzavara, S., Rabitti, A., and Bugliesi, M. Content security problems?: evaluating the effectiveness of content security policy in the wild. In Weippl, E. R., Katzenbeisser, S., Kruegel, C., Myers, A. C., and Halevi, S., editors, 2016.

[CYK$^+$18] Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N. L. U., John, R. S., Constant, N., Guajardo-Céspedes, M., Yuan, S., Tar, C., Sung, Y.-h., Strope, B., and Kurzweil, R. Universal Sentence Encoder. In *In Submission to: EMNLP Demonstration*, Brussels, Belgium, 2018.

[Cha17] Chapple, M. How expensive are ipsec vpn setup costs? 2017.

[COC+20] Chapuis, B., Omolola, O., Cherubini, M., Humbert, M., and Huguenin, K. An empirical study of the use of integrity verification mechanisms for web subresources. In, 2020.

[CCK+11] Cheikes, B. A., Cheikes, B. A., Kent, K. A., and Waltermire, D. *Common platform enumeration: Naming specification version 2.3.* US Department of Commerce, National Institute of Standards and Technology, 2011.

[CvRC+17] Chung, T., van Rijswijk-Deij, R., Chandrasekaran, B., Choffnes, D. R., Levin, D., Maggs, B. M., Mislove, A., and Wilson, C. A longitudinal, end-to-end view of the DNSSEC ecosystem. In, 2017.

[Cis] Cisco. Campus LAN and Wireless LAN Design Guide. Accessed: 08/12/2021. URL: https://www.cisco.com/c/en/us/td/docs/solutions/CVD/Campus/cisco-campus-lan-wlan-design-guide.pdf.

[DK21a] Dax, A. and Künnemann, R. On the soundness of infrastructure adversaries. In *2021 IEEE 34th Computer Security Foundations Symposium (CSF)*. IEEE, 2021.

[DK21b] Dax, A. and Künnemann, R. On the soundness of infrastructure adversaries. In, 2021.

[Del] Deloitte. Deloitte contractor site hourly rates. Accessed: 30/10/2018. URL: https://www2.deloitte.com/content/dam/Deloitte/us/Documents/public-sector/us-fed-contractor-site-hourly-rates-10172014.pdf.

[DH13] Dobson, S. and Haslum, P. Heuristics for bounded-cost search. In *Workshop on Heuristic Search and Domain Independent Planning (HSDIP'17)*, 2013.

[DOJ21] DOJ. Four chinese nationals working with the ministry of state security charged with global computer intrusion campaign targeting intellectual property and confidential business information, including infectious disease research, 2021. URL: https://www.justice.gov/opa/pr/four-chinese-nationals-working-ministry-state-security-charged-global-computer-intrusion.

[DH15] Dukhovni, V. and Hardaker, W. SMTP Security via Opportunistic DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS). RFC 7672, IETF, 2015.

[Dun12] Duncan, G., 2012.

[DLK+15a] Durkota, K., Lisy, V., Kiekintveld, C., and Bosansky, B. Game-theoretic algorithms for optimal network security hardening using attack graphs. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS'15)*, 1773–1774, 2015.

[DLK+15b] Durkota, K., Lisy, V., Kiekintveld, C., and Bosansky, B. Optimal network security hardening using attack graph games. In Yang, Q., editor, *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*. AAAI Press/IJCAI, 2015.

[DL14] Durkota, K. and Lisý, V. Computing optimal policies for attack graphs with action failures and costs. In *7th European Starting AI Researcher Symposium (STAIRS'14)*, 2014.

[DLB$^+$15] Durkota, K., Lisý, V., Bosanský, B., and Kiekintveld, C. Approximate solutions for attack graph games with imperfect information. In *Proceedings of the 6th International Conference on Decision and Game Theory for Security (GameSec'15)*, 228–249, 2015.

[DLK$^+$16] Durkota, K., Lisý, V., Kiekintveld, C., Bosanský, B., and Pechoucek, M. Case studies of network defense with attack graph games. *IEEE Intelligent Systems*, 31(5):24–30, 2016.

[EK09] Edelkamp, S. and Kissmann, P. Optimal symbolic planning with action costs and preferences. In [Bou09], 1690–1695.

[edg] edgescan. 2019 Vulnerability Statistics Report.

[eal21] Et al., S. R. 12 angry developers - A qualitative study on developers' struggles with CSP. In, 2021.

[Eur16] European Data Protection Supervisor. Privacy shield: more robust and sustainable solution needed, 2016.

[FKS14] Fett, D., Küsters, R., and Schmitz, G. An expressive model for the web infrastructure: definition and application to the browser ID SSO system. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, 673–688. IEEE Computer Society, 2014.

[FW20] Fila, B. and Wideł, W. Exploiting attack–defense trees to find an optimal set of countermeasures. In *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*, 395–410. IEEE, 2020.

[fNSA15] For NSA Cooperation, G. I. U. F. Maik baumgärtner and nikolaus blome and hubert gude and marcel rosenbach and jörg schindler and fidelius schmid, 2015.

[FER$^+$16] Frey, S., Elkhatib, Y., Rashid, A., Follis, K., Vidler, J., Race, N. J. P., and Edwards, C. It bends but would it break? topological analysis of BGP infrastructures in europe. In *IEEE European Symposium on Security and Privacy*, 2016.

[fKon14] Für Konsumforschung, G. Die große office-studie 2014, 2014.

[Gar15] Garner, P. Average revenue per user is an important growth driver, 2015.

[GLP05] Genesereth, M. R., Love, N., and Pell, B. General game playing: overview of the AAAI competition. *AI Magazine*, 26(2):62–72, 2005.

[GG09] Ghosh, N. and Ghosh, S. K. An intelligent technique for generating minimal attack graph. In *Proceedings of the 1st Workshop on Intelligent Security (SecArt'09)*, 2009.

[GmB16] GmBH, C. C. Studie zur mobilen e-mail-nutzung in deutschland, 2016. Survey took place in December 2015.

[Gol14] Goldberg, S. Why is it taking so long to secure internet routing? *Commun. ACM*, 57, 2014.

[Gre] Greenbone. OpenVAS - Open Vulnerability Assessment Scanner. URL: https://www.openvas.org/ (visited on 05/03/2020).

[Gre13] Greenwald, G. Xkeyscore: nsa tool collects 'nearly everything a user does on the internet'. *The Guardian*, July 31, 2013.

[GIL$^+$12] Gregori, E., Improta, A., Lenzini, L., Rossi, L., and Sani, L. On the incompleteness of the as-level graph: a novel methodology for BGP route collector placement. In, 2012.

[Gri13] Grigorik, I. *High Performance Browser Networking: What every web developer should know about networking and web performance.* O'Reilly Media, September 2013.

[GPS$^+$17] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On Calibration of Modern Neural Networks, August 3, 2017. arXiv: 1706.04599 [cs].

[HMN15] Halevi, T., Memon, N., and Nov, O. Spear-Phishing in the Wild: A Real-World Study of Personality, Phishing Self-Efficacy and Vulnerability to Spear-Phishing Attacks. SSRN Scholarly Paper ID 2544742, Social Science Research Network, Rochester, NY, January 2, 2015.

[Har15] Harding, L. Mass surveillance is fundamental threat to human rights, says european report. *The Guardian*, January 2015.

[Has13] Haslum, P. Heuristics for bounded-cost search. In [BFK$^+$13].

[HST12] Haufe, S., Schiffel, S., and Thielscher, M. Automated verification of state sequence invariants in general game playing. *Artificial Intelligence*, 187:1–30, 2012.

[Hel06] Helmert, M. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[Hel09] Helmert, M. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, 173:503–535, 2009.

[HD09] Helmert, M. and Domshlak, C. Landmarks, critical paths and abstractions: What's the difference anyway? In Gerevini, A., Howe, A., Cesta, A., and Refanidis, I., editors, *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, 162–169. AAAI Press, 2009.

[Hip09] Hippner, C. A Study into the Size of the World's Intelligence Industry. PhD thesis, Mercyhurst College, Pennsylvania, 2009.

[Hof15] Hoffmann, J. Simulated penetration testing: from "Dijkstra" to "Turing Test++". In Brafman, R., Domshlak, C., Haslum, P., and Zilberstein, S., editors, *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, 364–372. AAAI Press, 2015.

[HN01] Hoffmann, J. and Nebel, B. The FF planning system: fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[HYY$^+$18] Hong, G., Yang, Z., Yang, S., Zhang, L., Nan, Y., Zhang, Z., Yang, M., Zhang, Y., Qian, Z., and Duan, H. How you get shot in the back: a systematical study about cryptojacking in the real world. In, 2018.

[ICA21] ICANN. Tld dnssec report, 2021. Accessed: 01/08/2021. URL: http://stats.research.icann.org/dns/tld_report/.

[Inc17] Inc., G. Email encryption in transit, 2017.

[Inf17] Info Tech Research Group. Data Center & Facilities Optimization. https://www.infotech.com/research/ss/consolidate-data-centers, 2017.

[Ins16] Institute, S. I. P. R. Trends in world military expenditure, 2016. Retrieved 24 April 2017.

[Int16] Internet Live Stats. Internet users by country, 2016.

[JNO05] Jajodia, S., Noel, S., and O'Berry, B. Topological analysis of network attack vulnerability. In, *Managing Cyber Threats: Issues, Approaches and Challenges*, chapter 5. 2005.

[JPB$^+$19] Jangda, A., Powers, B., Berger, E. D., and Guha, A. Not so fast: analyzing the performance of webassembly vs. native code. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, 107–120, Renton, WA. USENIX Association, July 2019.

[JYD$^+$17] Jing, J. T. W., Yong, L. W., Divakaran, D. M., and Thing, V. L. L. Augmenting MulVAL with automated extraction of vulnerabilities descriptions. In *TENCON 2017 - 2017 IEEE Region 10 Conference*. TENCON 2017 - 2017 IEEE Region 10 Conference, 476–481, Penang. IEEE, November 2017.

[Jus16] Justin Dorfman, J. M. How to implement sri in your build process, 2016. Accessed: 01/03/2021. URL: https://hacks.mozilla.org/2016/04/how-to-implement-sri-into-your-build-process.

[Kam08] Kaminsky, D. Black ops 2008: it's the end of the cache as we know it. *Black Hat USA*, 2008.

[Kar08] Karasakal, O. Air defense missile-target allocation models for a naval task group. *Computers & Operations Research*, 35(6):1759–1770, 2008.

[KG09] Keyder, E. and Geffner, H. Soft goals can be compiled away. *Journal of Artificial Intelligence Research*, 36:547–556, 2009.

[KOE$^+$09] Kil, H., Oh, S., Elmacioglu, E., Nam, W., and Lee, D. Graph theoretic topological analysis of web service networks. *World Wide Web*, 12, 2009.

[KE11] Kissmann, P. and Edelkamp, S. Improving cost-optimal domain-independent symbolic planning. In Burgard, W. and Roth, D., editors, *Proceedings of the 25th National Conference of the American Association for Artificial Intelligence (AAAI'11)*, 992–997, San Francisco, CA, USA. AAAI Press, July 2011.

[KH14] Kissmann, P. and Hoffmann, J. BDD ordering heuristics for classical planning. *Journal of Artificial Intelligence Research*, 51:779–804, 2014.

[Kle08] Klensin, J. C. Simple mail transfer protocol. Internet RFC 5321, October 2008.

[KMW12] Kolobov, A., Mausam, and Weld, D. S. Discovering hidden structure in factored MDPs. *Artificial Intelligence*, 189:19–47, 2012.

[KVM$^+$18] Konoth, R. K., Vineti, E., Moonsamy, V., Lindorfer, M., Kruegel, C., Bos, H., and Vigna, G. Minesweeper: an in-depth look into drive-by cryptocurrency mining and its defense. In, 2018.

[Kon17] Konsult-center, 2017.

[KKM$^+$13] Kordy, B., Kordy, P., Mauw, S., and Schweitzer, P. ADTool: security analysis with attack-defense trees. In *Proceedings of the 10th International Conference on Quantitative Evaluation of Systems (QEST'13)*, 173–176, 2013.

[KMR$^+$10] Kordy, B., Mauw, S., Radomirovic, S., and Schweitzer, P. Foundations of attack-defense trees. In *Proceedings of the 7th International Workshop on Formal Aspects in Security and Trust (FAST'10)*, 80–95, 2010.

[KPS14] Kordy, B., Piètre-Cambacédès, L., and Schweitzer, P. Dag-based attack and defense modeling: don't miss the forest for the attack trees. *Computer science review*, 13:1–38, 2014.

[KW17] Kordy, B. and Wideł, W. How well can i secure my system? In *International Conference on Integrated Formal Methods*, 332–347. Springer, 2017.

[KW18] Kordy, B. and Wideł, W. On quantitative analysis of attack–defense trees with repeated labels. In *International Conference on Principles of Security and Trust*, 325–346. Springer, 2018.

[KYK$^+$11] Korzhyk, D., Yin, Z., Kiekintveld, C., Conitzer, V., and Tambe, M. Stackelberg vs. nash in security games: an extended investigation of interchangeability, equivalence, and uniqueness. *J. Artif. Int. Res.*, 41(2):297–327, May 2011.

[KHS$^+$18] Kulynych, B., Hayes, J., Samarin, N., and Troncoso, C. Evading classifiers in discrete domains with provable optimality guarantees. *arXiv preprint arXiv:1810.10939*, 2018.

[Lan13] Lancellotti-Young, C. The new gmail: +1 for revenue, -1 for opens, 2013.

[LKM07] Larbi, R. B., Konieczny, S., and Marquis, P. Extending classical planning to the multi-agent case: A game-theoretic approach. In *Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'07)*, 731–742, 2007.

[LGvR$^+$20] Lee, H., Gireesh, A., van Rijswijk-Deij, R., Kwon, T., and Chung, T. A longitudinal and comprehensive study of the DANE ecosystem in email. In, 2020.

[Lee13] Lee, T. B. Here's everything we know about prism to date, 2013.

[Lew17] Lewkowicz, K. 2017 state of email report, 2017.

[Lim14] Limited, D. S. C. How much are you worth? average revenue per user at google, facebook and twitter, 2014.

[LI05] Lippmann, R. P. and Ingols, K. W. An Annotated Review of Past Papers on Attack Graphs. Technical Report, Massachusetts Institute of Technology, 2005.

[LM] Llanso, T. and McNeil, M. Estimating Software Vulnerability Counts in the Context of Cyber Risk Assessments:7.

[Lou13] Louis, T. How much is a user worth? 2013.

[LHG08] Love, N. C., Hinrichs, T. L., and Genesereth, M. R. General Game Playing: Game Description Language Specification. Technical report LG-2006-01, Stanford Logic Group, 2008.

[LSR10] Lucangeli, J., Sarraute, C., and Richarte, G. Attack planning in the real world. In *Proceedings of the 2nd Workshop on Intelligent Security (SecArt'10)*, 2010.

[LAT$^+$20] Luceri, L., Andreoletti, D., Tornatore, M., Braun, T., and Giordano, S. Measurement and control of geo-location privacy on twitter. *Online social networks and media*, 17:100078, 2020.

[Lyo] Lyon, G. Nmap: the Network Mapper - Free Security Scanner. URL: https://nmap.org/ (visited on 05/03/2020).

[MPM$^+$17] Magnaguagno, M. C., Pereira, R. F., Móre, M. D., and Meneguzzi, F. Web planner: a tool to develop classical planning domains and visualize heuristic state-space search. In *Proceedings of the Workshop on User Interfaces and Scheduling and Planning, UISP*, 32–38, 2017.

[MWD$^+$15] Marczak, B., Weaver, N., Dalek, J., Ensafi, R., Fifield, D., McKune, S., Rey, A., Scott-Railton, J., Deibert, R., and Paxson, V. An analysis of china's "great cannon". In, 2015.

[MRL$^+$16] Margolis, D., Risher, M., Lidzborski, N., Chuang, W., Long, B., Ramakrishnan, B., Brotman, A., Jones, J., Martin, F., Umbach, K., and Laber, M. SMTP Strict Transport Security. Internet-Draft, Internet Engineering Task Force, March 2016. 20 pages. Work in Progress.

[Mar] MarketsandMarkets. Security Assessment Market Forecast 2022, CAGR 26.1% | Scope & Size. URL: https://www.marketsandmarkets.com/Market-Reports/security-assessment-market-128992392.html (visited on 04/01/2020).

[MO05] Mauw, S. and Oostdijk, M. Foundations of attack trees. In *Proceedings of the 8th International Conference on Information Security and Cryptology (ICISC'05)*, 186–198, 2005.

[Max17] MaxMind. IP Geolocation and Online Fraud Prevention. http://dev.maxmind.com/, 2017.

[McM93] McMillan, K. L. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[MDN21] MDN. Certificate transparency, 2021. Accessed: 23/07/2021. URL: https://developer.mozilla.org/en-US/docs/Web/Security/Certificate_Transparency.

[Met] Metasploit. Metasploit | Penetration Testing Software, Pen Testing Security. URL: https://www.metasploit.com/ (visited on 05/03/2020).

[NA10] Network, E. and Agency, I. S. The cost of dnssec deployment, 2010.

[NEJ+09] Noel, S., Elder, M., Jajodia, S., Kalapa, P., O'Hare, S., and Prole, K. Advances in topological vulnerability analysis. In *Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security (CATCH'09)*, 124–129, 2009.

[Nor14] Norddeutscher Rundfunk. Snowden-interview: transcript, January 2014.

[OCo17] O'Conner, T. German military battles foreign hacking with new cyber soldiers, 2017.

[OR20] Oesch, S. and Ruoti, S. That was then, this is now: a security evaluation of password generation, storage, and autofill in browser-based password managers. In *USENIX Security Symposium*, 2165–2182, 2020.

[OTG+16] Office, C., Treasury, H., Gummer, B., and Hammond, P. Britain's cyber security bolstered by world-class strategy, 2016.

[OPW+10] Oliveira, R. V., Pei, D., Willinger, W., Zhang, B., and Zhang, L. The (in)completeness of the observed internet as-level structure. *IEEE/ACM Trans. Netw.*, 18(1), 2010.

[Ope] OpenSCAP. Openscap. URL: https://www.open-scap.org/ (visited on 12/09/2021).

[OFS18] Orekondy, T., Fritz, M., and Schiele, B. Connecting pixels to privacy and utility: automatic redaction of private information in images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8466–8475, 2018.

[OKL+12] Osterweil, E., Kaliski, B., Larson, M., and McPherson, D. Reducing the x. 509 attack surface with dnssec's dane. *Securing and Trusting Internet Names, SATIN*, 12, 2012.

[OBM06] Ou, X., Boyer, W. F., and McQueen, M. A. A scalable approach to attack graph generation. In *ACM Conference on Computer and Communications Security*, 336–345, 2006.

[OGA05] Ou, X., Govindavajhala, S., and Appel, A. W. Mulval: a logic-based network security analyzer. In *Proceedings of the 14th Conference on USENIX Security Symposium - Volume 14*, SSYM'05, 8–8, Baltimore, MD. USENIX Association, 2005.

[Pee17] PeeringDB. De-cix frankfurt, 2017.

[Pen17] Penning, H. P. Analysis of the strong set in the pgp web of trust, 2017.

[PS98a] Phillips, C. and Swiler, L. P. A graph-based system for network-vulnerability analysis. In *Proceedings of the New Security Paradigms Workshop*, 1998.

[PS98b] Phillips, C. and Swiler, L. P. A graph-based system for network-vulnerability analysis. In *Proceedings of the 1998 Workshop on New Security Paradigms - NSPW '98*. The 1998 Workshop, 71–79, Charlottesville, Virginia, United States. ACM Press, 1998.

[PvGT$^+$19] Pochat, V. L., van Goethem, T., Tajalizadehkhoob, S., Korczynski, M., and Joosen, W. Tranco: A research-oriented top sites ranking hardened against manipulation. In, 2019.

[Pyt20] Python. Tldextract 3.1.0, 2020. URL: https://pypi.org/project/tldextract/.

[R A05] R. Arends and R. Austein and M. Larson and D. Massey and S. Rose. Resource Records for the DNS Security Extensions. RFC 4034, IETF, 2005.

[Ram16] Ramakrishnan, B. Measuring smtp starttls deployment quality, 2016.

[RS05] Ramasubramanian, V. and Sirer, E. G. Perils of transitive trust in the domain name system. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, 35–35, 2005.

[RGE$^+$16] Raumer, D., Gallenmüller, S., Emmerich, P., Märdian, L., and Carle, G. Efficient serving of vpn endpoints on cots server hardware. In *Cloud Networking*, 2016.

[Ren13] Rensfeldt, G. Se status in the intelligence community, 2013.

[Res00] Rescorla, E. HTTP Over TLS. RFC 2818 (Informational), IETF, 2000.

[RWX$^+$02] Rexford, J., Wang, J., Xiao, Z., and Zhang, Y. BGP routing stability of popular destinations. In *Proc. 2th ACM IMW Conference*, 2002.

[Hof02] Hoffman, P. SMTP Service Extension for Secure SMTP over Transport Layer Security. RFC 3207, RFC, IETF, 2002.

[BCN$^+$03] Barbir, A., Cain, B., Nair, R., and Spatscheck, O. Known Content Network (CN) Request-Routing Mechanisms. RFC 3568 (Informational), RFC, Fremont, CA, USA: RFC Editor, July 2003.

[KS05] Kent, S. and Seo, K. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, December 2005. Updated by RFCs 6040, 7619.

[Hou05] Housley, R. Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP). RFC 4309 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, December 2005.

[HS12] Hoffman, P. and Schlyter, J. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, August 2012.

[HJB12] Hodges, J., Jackson, C., and Barth, A. HTTP Strict Transport Security (HSTS). RFC 6797 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, November 2012.

[LLK13] Laurie, B., Langley, A., and Kasper, E. Certificate Transparency. RFC 6962 (Experimental), RFC, Fremont, CA, USA: RFC Editor, June 2013.

[FR14] Fielding, R. and Reschke, J. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. RFC 7231 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, June 2014.

[Mel16] Melnikov, A. Updated Transport Layer Security (TLS) Server Identity Check Procedure for Email-Related Protocols. RFC 7817, RFC, IETF, 2016.

[HSH19] Hallam-Baker, P., Stradling, R., and Hoffman-Andrews, J. DNS Certification Authority Authorization (CAA) Resource Record. RFC 8659, RFC, Fremont, CA, USA: RFC Editor, November 2019.

[RWH11] Richter, S., Westphal, M., and Helmert, M. LAMA 2008 and 2011 (planner abstract). In *IPC 2011 planner abstracts*, 50–54, 2011.

[RIP17a] RIPE Atlas. Internet data collection system. https://atlas.ripe.net/, 2017.

[RIP17b] RIPE Stat. Information about specific IP addresses and prefixes, 2017. URL: https://stat.ripe.net/.

[RIP] RIPE Stat. Information about specific IP addresses and prefixes. https://stat.ripe.net/.

[Ris17] RiskBasedSecurity. CVSSv3: New System, Next Problem (Scope). February 9, 2017. URL: https://www.riskbasedsecurity.com/2017/02/09/cvssv3-new-system-next-problem-scope/ (visited on 09/04/2019).

[RA00a] Ritchey, R. W. and Ammann, P. Using model checking to analyze network vulnerabilities. In *IEEE Symposium on Security and Privacy*, 156–165, 2000.

[RA00b] Ritchey, R. W. and Ammann, P. Using Model Checking to Analyze Network Vulnerabilities. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, SP '00, 156–, Washington, DC, USA. IEEE Computer Society, 2000.

[SBH12] Sarraute, C., Buffet, O., and Hoffmann, J. POMDPs make better hackers: accounting for uncertainty in penetration testing. In Hoffmann, J. and Selman, B., editors, *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI'12)*, 1816–1824, Toronto, ON, Canada. AAAI Press, July 2012.

[SC08] Scaparra, M. P. and Church, R. L. A bilevel mixed-integer program for critical infrastructure protection planning. *Computers & Operations Research*, 35(6):1905–1923, 2008.

[Sch16] Schiff, A. Facebook made almost 20 dollars in average revenue per user in q4 a big jump, 2016.

[Sch99] Schneier, B. Attack trees. *Dr. Dobbs Journal*, 1999.

[SPS$^+$17] Seipp, J., Pommerening, F., Sievers, S., and Helmert, M. Downward Lab. https://doi.org/10.5281/zenodo.790461, 2017.

[SPS$^+$16] Seipp, J., Pommerening, F., Sievers, S., and Wehrle, M. Fast Downward Aidos. In *UIPC 2016 planner abstracts*, 28–38, 2016.

[SSS14] Shandilya, V., Simmons, C. B., and Shiva, S. Use of Attack Graphs in Security Systems. 2014. URL: https://www.hindawi.com/journals/jcnc/2014/818957/ (visited on 03/31/2020).

[SHJ$^+$02] Sheyner, O., Haines, J. W., Jha, S., Lippmann, R., and Wing, J. M. Automated generation and analysis of attack graphs. In *IEEE Symposium on Security and Privacy*, 273–284, 2002.

[SJY$^+$14] Shieh, E. A., Jiang, A. X., Yadav, A., Varakantham, P., and Tambe, M. Unleashing Dec-MDPs in security games: enabling effective defender teamwork. In Schaub, T., editor, *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI'14)*, 819–824, Prague, Czech Republic. IOS Press, August 2014.

[Shi14] Shirman, J. Cloud, server migration and price elasticity, 2014.

[SW14] Shulman, H. and Waidner, M. Towards forensic analysis of attacks with dnssec. In *2014 IEEE Security and Privacy Workshops*, 2014.

[SJB$^+$14] Silver, D., Jana, S., Boneh, D., Chen, E., and Jackson, C. Password managers: attacks and defenses. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 449–464, 2014.

[SPR$^+$17a] Simeonovski, M., Pellegrino, G., Rossow, C., and Backes, M. Who controls the internet? analyzing global threats using property graph traversals. In *Proceedings of the 26th International Conference on World Wide Web*, 647–656, 2017.

[SPR$^+$17b] Simeonovski, M., Pellegrino, G., Rossow, C., and Backes, M. Who controls the internet?: analyzing global threats using property graph traversals. In *International Conference on World Wide Web*, 2017.

[Sop17] Sophos Ltd. Dedicated sophos email appliances, 2017.

[SGM20] Speck, D., Geißer, F., and Mattmüller, R. When perfect is not good enough: on the search behaviour of symbolic heuristic search. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS'20)*, 263–271. AAAI Press, 2020.

[Spe22] Speicher, P. Web-based interactive visualization of the case study for pareto-optimal defensive strategies for securing the web, 2022. URL: https://mitigation-web.github.io/.

[SSK$^+$17] Speicher, P., Steinmetz, M., Künnemann, R., Simeonovski, M., Pellegrino, G., Hoffmann, J., and Backes, M. Source code and interactive visualization of the results. 2017.

[SSK$^+$18b] Speicher, P., Steinmetz, M., Künnemann, R., Simeonovski, M., Pellegrino, G., Hoffmann, J., and Backes, M. Formally reasoning about the cost and efficacy of securing the email infrastructure. In, 2018.

[SST21] Speicher, P., Steinmetz, M., and Torralba, A. An extension of fast downward that implements the stackelberg planning algorithms, 2016-2021. URL: https://bitbucket.org/PatrickSpeicher/sim-pentest-what-if/.

[STT20] Speicher, P., Tran, N., and Torralba, A. Planning in the browser by compiling fast downward to webassembly, 2020. URL: https://project.cispa.io/fd-in-browser/.

[SSM10] Stamm, S., Sterne, B., and Markham, G. Reining in the web with content security policy. In, 2010.

[SBE$^+$21] Stan, O., Bitton, R., Ezrets, M., Dadon, M., Inokuchi, M., Ohta, Y., Yagyu, T., Elovici, Y., and Shabtai, A. Heuristic approach for countermeasure selection using attack graphs. In *2021 IEEE 34th Computer Security Foundations Symposium (CSF)*, 1–16. IEEE, 2021.

[Sta] Statcounter. Desktop browser market share worldwide. Accessed: 23-07-2021. URL: http://gs.statcounter.com/browser-market-share/desktop/worldwide.

[SMJ$^+$21] Steffens, M., Musch, M., Johns, M., and Stock, B. Who's hosting the block party? studying third-party blockage of csp and sri. In, 2021.

[SRJ$^+$19] Steffens, M., Rossow, C., Johns, M., and Stock, B. Don't trust the locals: investigating the prevalence of persistent client-side cross-site scripting in the wild. In, 2019.

[SHB16] Steinmetz, M., Hoffmann, J., and Buffet, O. Goal probability analysis in mdp probabilistic planning: exploring and enhancing the state of the art. *Journal of Artificial Intelligence Research*, 57:229–271, 2016.

[SFv$^+$14] Stern, R., Felner, A., van den Berg, J., Puzis, R., Shah, R., and Goldberg, K. Potential-based bounded-cost search and anytime non-parametric A*. *Artificial Intelligence*, 214:1–25, 2014.

[SJ14] Stock, B. and Johns, M. Protecting users against xss-based password manager abuse. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, 183–194, 2014.

[Tam11] Tambe, M. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned.* Cambridge University Press, 2011.

[TL00] Templeton, S. J. and Levitt, K. E. A requires/provides model for computer attacks. In *Proceedings of the Workshop on New Security Paradigms (NSPW'00)*, 31–38, 2000.

[Ten] Tenable, Inc. Nessus. URL: https://www.tenable.com/products/nessus (visited on 05/03/2020).

[TSF$^+$12] Thayer, J. T., Stern, R., Felner, A., and Ruml, W. Faster bounded-cost search using inadmissible estimates. In [BMS$^+$12].

[The15] The Radicati Group, I. Email statistics report, 2015-2019, 2015.

[The17] The Radicati Group, I. Email statistics report, 2017-2019, 2017.

[Thi10] Thielscher, M. A general game description language for incomplete information games. In Fox, M. and Poole, D., editors, *Proceedings of the 24th National Conference of the American Association for Artificial Intelligence (AAAI'10)*, 994–999, Atlanta, GA, USA. AAAI Press, July 2010.

[Tiz18] Tizio, G. D. Drive-by Download Attacks as a Stackelberg Planning Problem. Master's thesis, University of Trento, 2018.

[Toe15] Toews, B. Subresource integrity, 2015. URL: https://githubengineering.com/subresource-integrity/.

[TAK$^+$17] Torralba, Á., Alcázar, V., Kissmann, P., and Edelkamp, S. Efficient symbolic search for cost-optimal planning. *Artificial Intelligence*, 242:52–79, 2017.

[TLB18] Torralba, Á., López, C. L., and Borrajo, D. Symbolic perimeter abstraction heuristics for cost-optimal planning. *Artificial Intelligence*, 259:1–31, 2018.

[TSK$^+$21a] Torralba, Á., Speicher, P., Künneman, Steinmetz, M., and Hoffmann, J. Code, benchmarks, and data of faster stackelberg planning via symbolic search and information sharing. https://doi.org/10.5281/zenodo.4320574, 2021.

[TS20] Tran, N. and Speicher, P. Fork of fast downward which enables compilation to webassembly, 2020. URL: https://github.com/abwesend890/downward-in-the-browser.

[Tri16] Trimble, M. Server location, jurisdiction, and server location requirements, 2016.

[TS05] Tsiakis, T. and Stephanides, G. The economic approach of information security. *Computers & security*, 24(2):105–108, 2005.

[UDH$^+$20] Urban, T., Degeling, M., Holz, T., and Pohlmann, N. Beyond the front page: measuring third party dynamics in the field. In, 2020.

[Vah13] Vahabi, P., 2013.

[Val89] Valmari, A. Stubborn sets for reduced state space generation. In *Proceedings of the 10th International Conference on Applications and Theory of Petri Nets*, 491–515, 1989.

[vSta34] Von Stackelberg, H. *Market Structure and Equilibrium*. Springer-Verlag, 1934.

[W3C19] W3C. Webassembly core specification, 2019.

[WJR$^+$15] Wang, M., Jayaraman, P. P., Ranjan, R., Mitra, K., Zhang, M., Li, E., Khan, S. U., Pathan, M., and Georgakopoulos, D. An overview of cloud based content delivery networks: research dimensions and state-of-the-art. *Trans. Large-Scale Data- and Knowledge-Centered Systems*, 20, 2015.

[Web17] Webs, K. (F. Statistics, 2017. Retrieved: May 16 2017.

[WH12] Wehrle, M. and Helmert, M. About partial order reduction in planning and computer aided verification. In [BMS$^+$12].

[WH14] Wehrle, M. and Helmert, M. Efficient stubborn sets: generalized algorithms and selection strategies. In Chien, S., Do, M., Fern, A., and Ruml, W., editors, *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS'14)*. AAAI Press, 2014.

[WHA$^+$13] Wehrle, M., Helmert, M., Alkhazraji, Y., and Mattmüller, R. The relative pruning power of strong stubborn sets and expansion core. In [BFK$^+$13].

[WSL$^+$16] Weichselbaum, L., Spagnuolo, M., Lekies, S., and Janc, A. CSP is dead, long live csp! on the insecurity of whitelists and the future of content security policy. In, 2016.

[WBA$^+$16] Weinberger, J., Braun, F., Akhawe, D., and Marier, F. Subresource Integrity. Technical report, W3C, 2016. URL: http://www.w3.org/TR/2016/REC-SRI-20160623/.

[WLR14] Weissbacher, M., Lauinger, T., and Robertson, W. K. Why is CSP failing? trends and challenges in CSP adoption. In Stavrou, A., Bos, H., and Portokalidis, G., editors, *Proc. of RAID-14*, 2014.

[WAF$^+$19] Widel, W., Audinot, M., Fila, B., and Pinchinat, S. Beyond 2014: Formal Methods for Attack Tree-based Security Modeling. *ACM Computing Surveys*, 52(4):1–36, September 2019.

[XJS08] Xiuzhen Chen, Jianhua Li, and Shaojun Zhang. Study of generating attack graph based on privilege escalation for computer networks. In *2008 11th IEEE Singapore International Conference on Communication Systems*. 2008 11th IEEE Singapore International Conference on Communication Systems, 1213–1217, November 2008.

[YZZ14] Yuan, W., Zhao, L., and Zeng, B. Optimal power grid protection through a defender–attacker–defender model. *Reliability Engineering & System Safety*, 121:83–89, 2014.

[ZHR$^+$18] Zhang, Y., Humbert, M., Rahman, T., Li, C.-T., Pang, J., and Backes, M. Tagvisor: a privacy advisor for sharing hashtags. In *Proceedings of the 2018 World Wide Web Conference*, 287–296, 2018.

[Zij11] Zijin, G. Analysis of chinese internet users from the leak, 2011.