Pareto-Optimal Defenses for the Web Infrastructure: Theory and Practice

GIORGIO DI TIZIO*, University of Trento, Italy
PATRICK SPEICHER*, CISPA Helmholtz Center for Information Security, Germany
MILIVOJ SIMEONOVSKI, Independent researcher, Germany
MICHAEL BACKES, CISPA Helmholtz Center for Information Security, Germany
BEN STOCK, CISPA Helmholtz Center for Information Security, Germany
ROBERT KÜNNEMANN, CISPA Helmholtz Center for Information Security, Germany

The integrity of the content a user is exposed to when browsing the web relies on a plethora of non-web technologies and an infrastructure of interdependent hosts, communication technologies, and trust relations. Incidents like the Chinese Great Cannon or the MyEtherWallet attack make it painfully clear: the security of end users hinges on the security of the surrounding infrastructure: 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 have 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 this work, we propose a graph-based 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 hacker groups to nation-state actors. 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. We find a considerable increase of security (up to 13% protected web visits) is possible at relatively modest cost, due to the effectiveness of mitigations at the application and transport layer, which dominate expensive infrastructure enhancements such as DNSSEC and IPsec.

CCS Concepts: • **Security and privacy** → *Formal security models*; *Domain-specific security and privacy architectures.*

Additional Key Words and Phrases: Formal security models, Security architectures, Web security

1 INTRODUCTION

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 website of their bank website, redirected to an exploit kit by means of drive-by download attacks,

*Both authors contributed equally to this research.

Authors' addresses: Giorgio Di Tizio, University of Trento, Via Sommarive 9, Trento, Italy, 38123, giorgio.ditizio@unitn.it; Patrick Speicher, patrick.speicher@cispa.de, CISPA Helmholtz Center for Information Security, Stuhlsatzenhaus 5, Saarbrücken, Germany, 66123; Milivoj Simeonovski, Independent researcher, Germany; Michael Backes, CISPA Helmholtz Center for Information Security, Stuhlsatzenhaus 5, Saarbrücken, Germany, 66123; Ben Stock, CISPA Helmholtz Center for Information Security, Stuhlsatzenhaus 5, Saarbrücken, Germany, 66123; Robert Künnemann, CISPA Helmholtz Center for Information Security, Stuhlsatzenhaus 5, Saarbrücken, Germany, 66123, robert.kuennemann@cispa.de.

2471-2566/2022/10-ART $15.00
https://doi.org/10.1145/3567595

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. Even the mechanisms to ensure confidentiality and availability like TLS rely on a 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 paper, 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 [66]), name resolution level (DNSSEC [45]), website delivery (HTTPS [74], HSTS [27]), public-key infrastructure (certificate transparency [13], DANE [61]), and third-party JS inclusions (CSP [4], SRI [29]). 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? A recent methodology to answer these questions was proposed as *Stackelberg planning* [43] in the AI community, 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 himself is combining attack actions to compromise specific targets. This method has been successfully applied to the security of the email infrastructure [42], where the application layer is much simpler and a large part of the population is typically served by a handful of email providers. By 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 planning problem, based on the graph database system Neo4J[1]. We represented Web entities (domains, NSs, ASes, etc.) and their relationships using a property graph and exploited Neo4j reachability queries to compute attack graphs and determine impact and cost of different mitigations. 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 success in a given scenario. With that, we can scale up to about 71k infrastructure elements and 2.2M attacker actions.

Our second contribution is a 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 with respect to the infrastructure.

Our third contribution is an analysis of these mitigation strategies securing clients that visit the Alexa Top 5k. We consider three classes of attackers in terms of their capabilities and initial asset: cyber-criminal groups (e.g. [81]), malicious infrastructure providers for cloud services and name resolutions, and nation-state attackers. For each, we compute the cost and the efficacy (in terms of reduction of the number of affected visits on the Top5k) of the existing deployments of that technology, and the Pareto-optimal deployment of defenses that

[1]https://neo4j.com/

balance cost and efficacy. We created a web-based GUI at mitigation-web.github.io to analyze and investigate optimal mitigation deployments with customizable costs.

Goals. 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 from mass attacks. We analyze the effect of mitigations that can avoid or limit the attacker's ability to affect user visits on websites. Our goal is to provide a framework to identify criticalities for the security of the Web due to dependencies on countries and infrastructure providers and help determine the mitigations that should be implemented as policies.

Non-goals. We do not focus on the greater goal of the attacker. The results of exploiting weaknesses of the Internet infrastructure can range from cryptojacking to phishing, attacks against password managers, or DDoS [17, 20, 24, 46, 70, 79] depending on the attacker's motivation and falls outside the scope of this paper. We do not consider targeted attacks on specific individual hosts, software, or companies nor provide ad hoc defenses for targeted individuals. The discussion of the incentives that can lead to the application of the mitigations as global policies are outside the scope of the paper.

2 STACKELBERG PLANNING

Planning is an area of AI dedicated to general-purpose mechanisms that automatically find a *plan*, when given a high-level description of the (relevant part of) the world properties: *propositions*, an *initial state*, and a *goal* condition (see [35] for an introduction). A plan is a sequence of *actions*, each described in terms of a *precondition* and a *postcondition*, from the initial state to a state that fulfills the goal condition.

Speicher et. al. proposed *Stackelberg planning*, which can be seen as a two-fold classical planning task [43]. Inspired by work on Stackelberg security games, the defender (leader) moves first and the attacker (follower) can fully observer the defender's action and can plan their best response accordingly. In our notion of Stackelberg planning, 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.*

For technical completeness, we recall Speicher et al.'s definition of Stackelberg planning tasks, which follows the classical STRIPS framework for planning. The discussion up to § 6, however, avoids using mathematical notation, hence the reader can feel free to skip this definition.

Definition 1 (Stackelberg planning (cited from [43], with slightly different terminology)). *A Stackelberg planning task is a tuple* $\Pi = (\mathcal{P}, \mathcal{A}^D, \mathcal{A}^A, \mathcal{I}, \mathcal{G}^A)$ *of a set of* facts $\mathcal{P}$*, a set of* defender actions $\mathcal{A}^D$*, a set of* attacker actions $\mathcal{A}^A$*, an* initial state $\mathcal{I} \subseteq \mathcal{P}$*, and the* attacker goal $\mathcal{G}^A \subseteq \mathcal{P}$*. We require that* $\mathcal{A}^D \cap \mathcal{A}^A = \emptyset$*, and we denote by* $\mathcal{A} = \mathcal{A}^D \cup \mathcal{A}^A$ *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* $\langle a_1, \ldots, a_n \rangle$ *is applicable in a state s if* a_1 *is applicable in s, and* $\langle a_2, \ldots, a_n \rangle$ *is applicable in* $s[[a_1]]$*. The resulting state is denoted* $s[[\langle a_1, \ldots, a_n \rangle]]$*. The cost of an action sequence is* $c_{\langle a_1, \ldots, a_n \rangle} = \sum_{i=1}^{n} c_{a_i}$*.*

A defender strategy *is a sequence of defender actions* π^D *applicable in* $\mathcal{I}$*. We denote by* $S^D \subseteq S$ *the set of all states reachable from* $\mathcal{I}$ *through a defender strategy. Let* $s \in S^D$ *be any such state. The defender's* best move *to s is a defender strategy* π^{D*} *to s whose cost is minimal among all defender strategies ending in s; we denote that cost by* $D^*(s)$*. A* attacker strategy *in s is a sequence of attacker actions* π^A *that is applicable in s, and that achieves the*

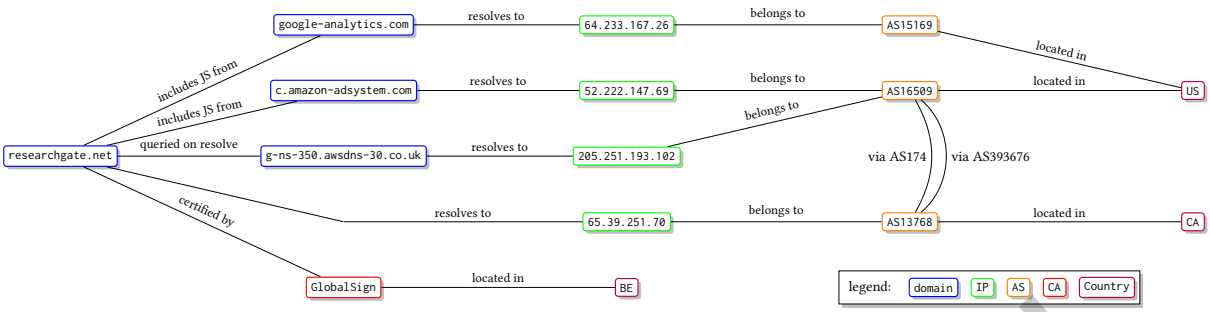


Fig. 1. A fragment of the infrastructure relations while visiting `researchgate.net`

attacker goal, i.e., , $\mathcal{G}^A \subseteq s[[\pi^A]]$. *The attacker's* best response *in s is a attacker strategy π^{A*} in s with minimal cost; we denote that cost by $A^*(s)$, or $A^*(s) = \infty$ if no attacker strategy exists.*

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

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. 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 §3 and §4 is a Stackelberg planning task, but instead of using Speicher et. al.'s algorithm to compute the solution (i.e., the Pareto frontier), we developed a graph-based algorithm (see §6).

3 THREAT MODEL

In this paper we focus on infrastructure attacks, *i.e.,* those that arise from physical, logical, and administrative dependencies in the Internet, as opposed to weaknesses in protocol 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 1 and formally defined in Appendix A. We define an entity (e.g. a NS, a route between ASes, etc.) in our model as compromised if the attacker is able to affect the integrity of the entity. For example, a route between two ASes is compromised if an attacker is able to pose as a MITM in the communication. Similarly, a NS is compromised if the attacker is able to tamper with the DNS response. 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 the attacker's actions lead to a loss of confidentiality and integrity for the users visiting the websites. For example, an attacker that is able to

Table 1. Attacker actions associated to class of attackers (nation-state (N), service providers (S), small hacker groups (H)), paraphrased (formal definition in Appendix A). *Mitigation due to sneakiness assumption.

	#	attack vector	precondition	outcome	applicable mitigations	attacker class
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 compr.	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 compr.	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 compr.	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 cert. 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 compr.	HTTPS + HSTS + HTTPS-Redirect (unless cert. 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)	third-party JS-inclusion	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

perform a MITM attack can both actively inject/modify content (loss of integrity) or passively eavesdrop on the communication (loss of confidentiality).

3.1 Infrastructure

To illustrate the most important infrastructure dependencies in the web, consider the following common place example (Fig. 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.

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, websites, CAs, or NSs that translate into a subset of rules described in Table 1. Not all of the attack vectors are available to all classes of attackers as some are traits specific to particular attackers. In Table 1 we identified which classes hold the capability for each attack vector. This will be used in the analysis in § 7. 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 visits to these web sites, i.e., the attacker plan maximizing $\sum_{i \in countries} Visits_{i,d}$ for $Visits_{i,d}$ being the estimated number of visits for the web site d from the Country i.[2]

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

For the class of attackers considered, the stealthiness is of the utmost importance to avoid attribution and retaliation [11], 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'. We discuss the limitations in §3.4.

3.3 Attacker Rules

The threat model is described in terms of attacker actions that are instances of the rules in Table 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.

[2] We use the number of visits 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.

- 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 Appendix A 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 [17]. Suppose users visit the popular website `www.diply.com`. By rule (1), 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)). The injected JS can force visitors to perform a DDoS attack against target websites [17] or to redirect the victims to a malicious web server to download malware.

3.4 Limitations

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* [9], but we consider this out of the scope and take this attacker model as granted. Moreover, their results suggest a tight relation between rules in the attacker model and protocol-level security properties. On the one hand, this gives guidance for the formulation of new attack vectors. On the other hand, precisely describing protocol-level security properties is known to be difficult and often done in conjunction with verification. 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 §5.1 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 in §7 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.

The threat model focuses on attacks that can lead an attacker to compromise the content of a web site as a result of physical and logical dependencies. Our model can be extended to describe additional web attacks, e.g.

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

vulnerable libraries or server misconfigurations, that produce a *direct compromise* of the content of a web site with a structure similar to rule (5) for XSS without impacting the methodology discussed in §6. We leave for future works this extension.

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. A mitigation can present some preconditions to be met before the deployment (*e.g.,* DANE requires DNSSEC, Certificate Transparency requires the presence of HTTPS, etc.). In Appendix A we formally defined the preconditions for each mitigation. In our analysis we allow a mitigation to be deployed only if its preconditions hold.

We gather the cost based on publicly available 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. For example, `youtube.com` and `google.com` belong to the same company, but only recently announced they will share infrastructure [1]. We detail our cost model in § 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.

4.1 Application layer mitigations

SRI. CDNs are a major 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 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 [2, 17]. The implementation of SRI for the resources retrieved from Baidu could have reduced the impact of this attack [84].

Although the adoption of SRI is growing [16], 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 [37].

Other mitigations. Another mitigation could be Content Security Policies (CSP) [49]. 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 [37], with the result that most of the deployments are insecure and enable inline scripts [34, 38, 50]. Given that CSP is mainly used to prevent inline XSS [38, 48] 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 § 9.

4.2 Transport Layer Mitigations

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.

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

[5]Only the most recent version of Chrome [5] and Firefox in private mode [6] use HTTPS by default. Safari defaults to HTTP.

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 [27]. 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]

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.

4.3 Routing Layer Mitigations

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 a transport-level gateway-to-gateway tunnel. Various technologies provide this functionality, but to provide a concrete cost estimate [7], we chose IPsec with a gateway-to-gateway architecture [62, 66].

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.

4.4 Resolution mitigations

DNSSEC. To prevent DNS spoofing attacks, DNS records can be authenticated with DNSSEC [45]. The adoption by end users is still very low [3], but it can be implemented in the recursive resolver of the ISP [69]. 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 (§7), 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 [63].

4.5 CA mitigations

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) [13] 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

[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

Table 2. Mitigation cost per host. Let $r = 140\,\$/h$ the daily rate of an external consultant.

Mitigation	Cost per host	Comment
SRI	r*8h	Consultant cost for 1 day. Exists tools to support (*e.g.,* [64]). 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 [37].
TLS	r*8h	Consultant cost for 1 day to modifying 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 [18] and the consultant cost for configuration and maintenance per year (about 80 consulting hours) [7].
DNSSEC	$366,342	Cost of deploying in all the authoritative NS managed by a company based on the maximum CAPEX from a survey [69], 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 [42]. Exists tools to automatically generate TLSA records (*https://ssl-tools.net/tlsa-generator*).

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 [78], and that Mozilla is planning to include support for the CT project [68], we assume the entire CA ecosystem to be CT compliant. Given that our threat model (§ 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.

Other mitigations. DNS-Based Authentication of Named Entities (DANE) [61] is a DNS-level mitigation against vulnerabilities in the CA model [23]. 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. This mitigation requires DNSSEC to be deployed. 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.

Although the adoption of DANE for email servers is growing, there are challenges that prevent the adoption for the Web PKI [25]. 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 §7.

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 [41]. We thus ignored this mitigation.

4.6 Mitigation Cost

We focus on the immediate cost of mitigations and convert all personnel cost from time estimates into $ by considering the cost for an external consultant[8] of $r = 140\,\$/\text{h}$. Table 2 lists the cost estimates. These values will be used to compute Pareto optimal defenses in §7. The investigation of optimal mitigation deployments with custom costs can be performed on our website mitigation-web.github.io.

4.7 Limitations

Like our threat model, our mitigations inherently focus on protocol-level attacks (§ 3).

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 [43], 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.

5 IMPLEMENTATION

In this section we discuss the data acquisition and the pre-processing to solve the resulting Stackelberg problem.

5.1 Data acquisition

Our mitigation analysis is performed on the Top 5k Alexa domains obtained from Tranco [56] 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. [36]

Web server data. We collected the `strict-transport-security` and `CSP` security headers 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.

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[9]. As the content of the landing page and its internal pages likely differ [57], to avoid the risk of capturing a limited subset of third-parties [55], 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 [73] 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.

[8]Hourly rate (US) for a Computer Security System Specialist level 2 via Deloitte, Ltd [10].
[9]We used the *Selenium Web* driver, an object-oriented API for web-app testing

DNS data. For each website and CDN, we collect the list of authoritative NSs 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 [54] (e.g. expired signatures), we did not investigate these issues.

Routing Information. To model the internet connectivity, *i.e.,* connectivity between ASes, we collect traceroutes provided by RIPE Atlas [75] 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.

Geolocation. For each NS, website, and CDN, we include their geolocation in our dataset. We link IPs to ASes using the RIPEstat database service [76] and we map ASes to countries using the MaxMind database [67]. In addition, each CA is mapped to a specific country using the information stored in the issuer section of the digital certificate.

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 collect the Country (C) field.

Limitations & Caveats

Visitors–To calculate the attacker reward (see § 3), we collected statistics about the number of visits on each domain using Alexa's `UrlInfo` API.
Routing Information–Achieving a global and complete routing coverage, where all possible routes between ASes are covered is an infeasible task, if not impossible [22, 47]. 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 [75] networks at the beginning of 2021[10].

5.2 Stackelberg Planning via Graph Databases

Speicher et. al [43] derived a general-purpose algorithm for Stackelberg planning, which was successfully applied to the security of the email infrastructure [42]. Their 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 case [83] with a threat model similar to ours. His thesis found, 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 that allows one to store data in the form of a property graph, i.e. a graph with different types of nodes and edges, and easily query the database by exploiting the relationships between nodes. As discussed in §3.1, the security of Web relies on relationships among several entities. The Web infrastructure (e.g. Fig. 1) naturally maps into this type of representation. From the Neo4j property graph and the set of rules of our model, instead of enumerating all relevant attacker actions in an input file (like in [42, 83]), 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 actions via the removal of edges. Neo4j's data structures are optimized for such queries. Moreover, by representing the data presented in

[10]We assume that BGP routes are reasonably stable [28].

§ 5.1 as a property graph, we drastically improve the generation time of this attack graph. In the follow-up, we provide a formal overview of our graph-based analysis.

6 GRAPH-BASED ANALYSIS

We use Neo4j to store and analyze a larger set of domains. In contrast to the fine-grained deployment analysis via the Stackelberg planning algorithm, 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.

Fig. 2 summarizes the procedure to perform a graph-based Stackelberg planning analysis. We start from the property graph describing the entire Web relationships. Starting from the attacker's initial assets we apply the rules of the model, that describe the attacker's actions, to generate an attack graph for the scenario. Finally, we apply a set of mitigations to remove some edges of the attack graph. We then queried the resulting attack graph using Neo4j to determine the reachability of the domains from the attacker node.

The attack graph is a directed graph with each node corresponding to a fact in the Stackelberg planning task and two nodes being connected if there is an attacker action that allows adding the latter (and only the latter) if the former is present. The initial assets are facts, and thus the graph would have multiple roots, however, to simplify reachability queries, we opted for a special root node `attacker` that connects to all initial assets and is the only node that is not a fact. The leaf nodes are the set of 'website compromised' nodes that are reachable when no mitigations are deployed, so that by removing edges, we can evaluate the impact of a countermeasure on the reachability of the leafs starting from `attacker`.

Note the difference to attack trees [77], where the root nodes describe a single goal and the parent-child relationship between subgoals can either be a conjunction (meaning all subgoals need to be achieved to reach the parent goal) or a disjunction. Our attack graphs are closer to Philips and Schwiler's attack graphs [71], where nodes describe the state of an attacker to a single goal. By contrast, we consider many goals and the state of the attacker corresponds to the set of nodes on the path(s) to the goal(s).

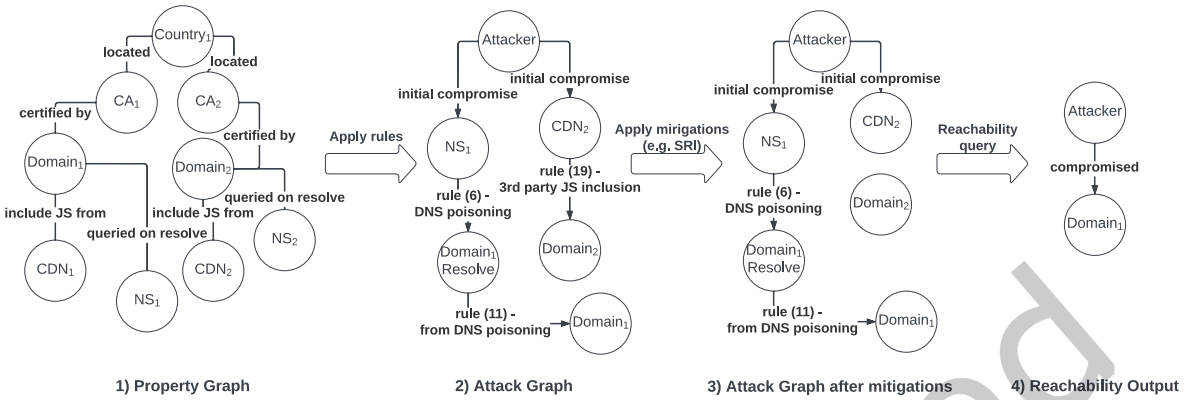
6.1 Notation

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 post(r) and preconditions *pre*(r). For instance, to describe all actions that compromise a website via XSS, we could use the following action schema with the variable x in post and precondition.

$$\frac{XSS(x)}{C^{\text{web}}(x)}$$

By giving x a domain with n values, this schema can be grounded, resulting in n actions. Our rules (discussed in detail in Appendix A) slightly extend this notation. Variables are always defined over the nodes of a property graph PG $= (V, E, l)$, i.e., their domain is V. Each vertex and edge in this graph are labeled with one or more labels from an arbitrary set L, hence the type of the labeling function $l : V \uplus E \to 2^L$.

In addition to precondition and postcondition, a rule r has a graph's property graph(r) which is a conjunction of atoms '$v \in z$' (v is labelled z) and '$v_1 \xrightarrow{z} v_2$' (v_1, v_2 are connected with an edge labelled z). For example, we can



From the Neo4j property graph, that describes the Web Infrastructure, we generate an attack graph for an attack scenario based on the initial assets of the attacker and the rules of the model. A combination of mitigations is applied to the attack graph to disable edges. Finally, a reachability query in Neo4j is performed to determine the domains reachable from the attacker node.

Fig. 2. Graph-based analysis for Stackelberg planning

limit the (value) domain of x to the set of (network) domains:

$$\frac{\overbrace{x \in D}^{\text{graph}(r)} \quad \overbrace{XSS(x)}^{pre(r)}}{C^{\text{web}}(x)}$$

Using Neo4J, we can efficiently evaluate these properties and find all satisfying assignments from variables to nodes in the graph. We define the semantics of graph properties as follows: let $\text{PG}, \sigma \vdash \phi$ denote that an assignment $\sigma : \mathcal{V} \to V$ satisfies a graph constraint ϕ on a property graph $\text{PG} = (V, E, l)$. Formally, $\text{PG}, \sigma \vdash \phi$ iff

$$\begin{aligned}
\text{PG}, \sigma \vdash x \in z &\iff \sigma(x) \in \{v \in V \mid l(v) = z\} \\
\text{PG}, \sigma \vdash x \xrightarrow{z} y &\iff (\sigma(x), \sigma(y)) \in \{e \in E \mid l(e) = z\} \\
\text{PG}, \sigma \vdash \phi_1 \wedge \phi_2 &\iff \text{PG}, \sigma \vdash \phi_1 \text{ and } \text{PG}, \sigma \vdash \phi_2
\end{aligned}$$

6.2 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 (Fig. 3). 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.

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.[11] In the resulting attack graph AG, a node represents an instantiation of a predicate and an edge represents an instantiation of a rule.

Algorithm 1: property graph to attack graph

Input: property graph PG, initial assets attacker
Output: attack graph AG
`// initialize AG`
1 AG $\leftarrow (V \cup \{\texttt{attacker}\}, \{(\texttt{attacker}, v) \mid v \in V\})$ with $V = \{C(x) \mid x \in \textit{initial assets attacker}\}$;
`// add compromised nodes`
2 **for** $r_i \in [1, 2]$ **do**
3 AG $\leftarrow$ AG + $\{\sigma(v) \rightarrow \sigma(w) \mid \text{PG}, \sigma \vdash \text{graph}(r_i) \wedge v \in \textit{pre}(r) \wedge w \in \text{post}(r)\}$;
`// apply (3) and (4) until fixpoint is reached`
4 **while** *fixpoint not reached* **do**
5 AG $\leftarrow$ AG + $\{\sigma(v) \rightarrow \sigma(w) \mid \text{PG}, \sigma \vdash \text{graph}(r_3) \wedge v \in \textit{pre}(r) \wedge w \in \text{post}(r)\}$;
6 AG $\leftarrow$ AG + $\{\sigma(v) \rightarrow \sigma(w) \mid \text{PG}, \sigma \vdash \text{graph}(r_4) \wedge v \in \textit{pre}(r) \wedge w \in \text{post}(r)\}$;
`// 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 $\leftarrow$ AG + $\{\sigma(v) \rightarrow \sigma(w) \mid \text{PG}, \sigma \vdash \text{graph}(r_i) \wedge v \in \textit{pre}(r) \wedge w \in \text{post}(r)\}$;
`// add mitigations`
9 mark all edges in AG as removable if a defender rule applies to it and they have no mitigation disabling dependency ;
10 **for** *ca* $\in$*{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;

6.3 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. Alg. 1 is used to translate a property graph into an attack graph. We provide a general version in Alg. 2 in Appendix B.

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 asset for the class of attackers (i.e. country for nation-state, AS and (optionally) CA for service providers, and domains/NSs for small hacker group), it traverses every rule in topological order w.r.t. the dependency graph (lines 2 and 7). For convenience, we introduce a starting note `attacker` that is connected to all the initial assets (line 1). 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

[11] 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.

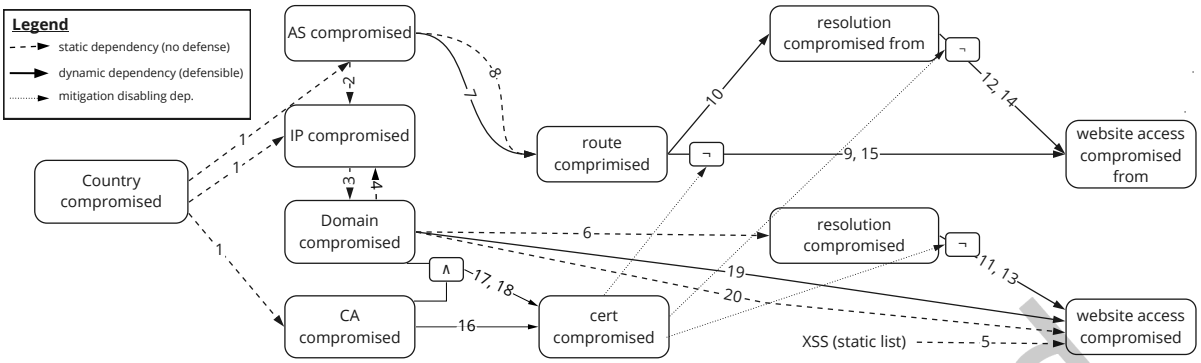


Fig. 3. Dependency graph: the rules impose a hierarchy on the compromise predicates. A dashed edge indicates a static dependency, a solid edge a dynamic, i.e., defensible, dependency and a dotted edge an attacker action that can disable a mitigation.

consisting of (3) and (4). The resulting graph AG after line 8 contains all attacker plans as paths starting from some 'initially-compromised' node.

6.4 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 preconditions, the application of a mitigation corresponds to the addition or 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[12] 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, $n = 9$). 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.

[12] Our specific mitigation schemas (see § 4 and Appendix A.2) can be ordered such that, if one mitigation schema depends on the other, the former appears before the latter. This holds, e.g., for the order in which they appear in Appendix A.2. Hence we can treat them as sets, otherwise, they need to be considered as lists. In both cases, the number of combinations grows exponentially.

They are disabled (marked with the ¬-symbol in Fig. 3) if the corresponding mitigation was selected and the 'CA compromised' predicate preventing the mitigation is not reachable for the attacker.

6.5 Applicability to other problems

The graph-based approach trades off generality for speed. As argued in § 5.2, it improves the scalability by two orders of magnitude when measured in target domains. Although we designed this algorithm specifically for the problem at hand, it can apply to many problems in this domain. We will describe the class of problem as a special class of Stackelberg problems.

First, the Stackelberg problem can be described using a property graph and action schemas with variables for nodes and graph preconditions as in § 6.1. Second, the attacker action schemas have

(1) only positive pre and post conditions (graph conditions can be negative),
(2) a single postcondition and
(3) the dependency graph is (largely) acyclic.

The third condition is technically not needed, as cycles can be resolved by fix-point computation, but this produces a performance overhead. Our technique is most effective if cycles are small and only a few iterations are necessary to reach the fix point. Third, the defender's actions must be expressible as the removal of edges. In particular, they cannot enable new actions for the attacker (which is very unusual in attacker defender games, but there are other applications for Stackelberg planning). A sufficient criterion is the following:

(1) Defender actions have only graph properties in preconditions and their postcondition is a single positive defender proposition.
(2) In attacker actions where defender propositions can only occur in preconditions, they only occur negatively, and only one at a time.

Fourth, the attacker's goal is to maximize the weighted sum of the set of propositions in the final state, while the defender tries to minimize the same sum, as well as the cost of the mitigation actions. As we simply enumerate all subsets of mitigations, the mitigation can be any computable function on the attacker state. Examples for problems in this space is the previous analysis of the email infrastructure [42] or any kind of tainting-based reachability analysis on graphs (e.g., [39]). Intuitively, we can think of an attacker that tries to obtain assets that either help in capturing other assets or are valuable by themselves. Assets may help and can never inhibit capturing another asset. Defensive measures disable attacker actions.

The limitation of the approach is its focus on graph-based analysis: fix point computations are costly and thus to be avoided. The approach scales well with the size of the graph (due to quick evaluation of graph queries) but poorly with the number of mitigations (as writing to the graph database is expensive). Moreover, the classical Stackelberg planning algorithm can prune some mitigation strategies, e.g., when a cheaper, more effective strategy was explored already, and thus avoid unnecessarily attacker runs. We leave the integration of this feature for future work.

7 EVALUATION

From the data collected in § 5.1 we evaluated attacks on the Web carried by the different class 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 1).

For each attack scenario, we generated the attack graph. We then ran the analysis on every combination of the mitigation strategies introduced in § 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 Tab. 3 and 4, 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 (§ 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.

The resulting Pareto frontiers depend on the costs discussed in §4.6. 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. Stakeholders can modify the cost assigned to all countermeasures or determine the interval of costs in which the Pareto frontier does (not) change.

Cyber-criminal Group. In this threat scenario, shown in the leftmost column of Table 4, we consider a hacker 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 Fig. 4 for the Pareto frontier). We took inspiration from the MyEtherWallet attack on the 24th of April 2018 [81] to evaluate the impact of an attack performed by cyber-criminal groups on the Web infrastructure. The original 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. The initial asset is thus composed of more than 1500 Amazon's *Route 53* NS. 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.[13] The most effective countermeasure is to employ secure connections both for the website 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 [81]. In summary, enforcing endpoint level defense is the optimal solution for this threat.

Infrastructure Providers. Next, we 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. The overall attacker success for each of the companies is shown in Table 3. Figure 5 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. The initial asset is generated starting from the owned ASes and CAs (if any) for each infrastructure provider. Amazon, Google, and GoDaddy control certification authorities that are accepted trust anchors in all major browsers. While CloudFlare

[13]For the actual BGP-based attack, the attacker cannot sign in the nameserver's stead, hence DNSSEC/DANE could appear as a possible mitigation in the high-cost range of the Pareto frontier

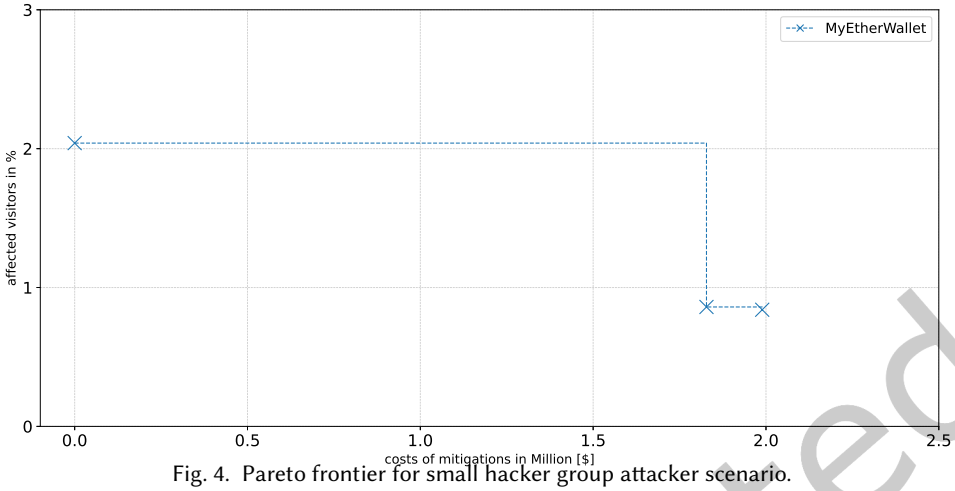


Fig. 4. Pareto frontier for small hacker group attacker scenario.

Table 3. 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$

and Dyn control ASes, but no CA. By propagating the rules (2),(3),(4) we compute the entire asset for the attack scenario. For example for Dyn, we further add its 175 NSs serving about 576 domains in our dataset. In Table 3, Google is the strongest player, affecting about 38% of the page views on the Top 5K Alexa domains. While Dyn only affects roughly 8% of the visits. In terms of attack vectors, we observed that Google has a great impact on

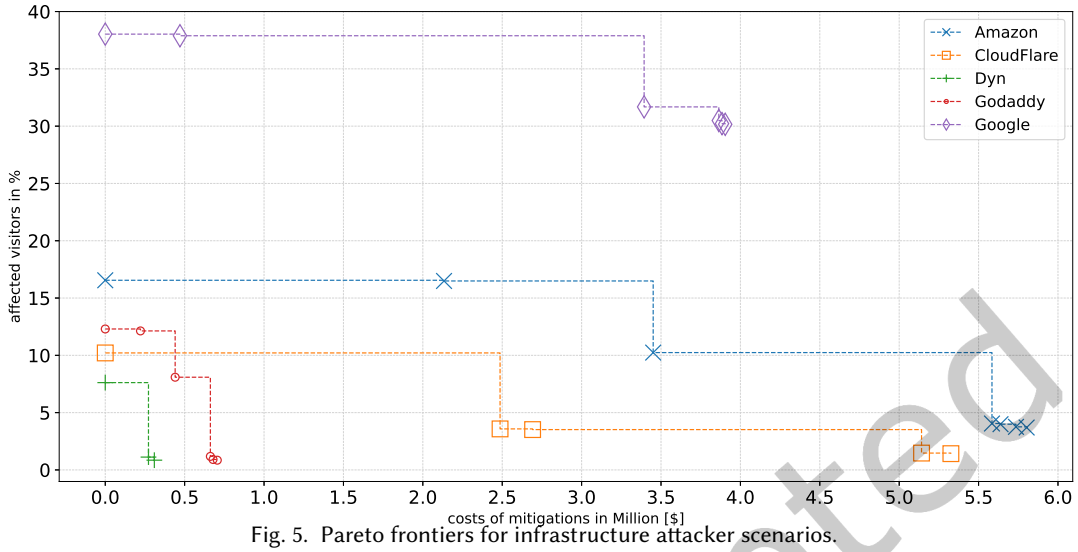


Fig. 5. Pareto frontiers for infrastructure attacker scenarios.

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.

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 3), 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 4. 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$

Nation-State Groups. In this scenario (Table 4, right-side columns), we consider the potential of three states to mount an attack, assuming that local legislation permits such an attack (see Fig. 6 for the Pareto frontier for each country). The Great Cannon attack, e.g., is believed to have been mounted from China. The initial asset is obtained starting from rule (1) by including the NSs, ASes, domains, and IPs located in that country.

The US is the country with the highest attack potential: about 46% of the visits on the Top5K 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 4, 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).

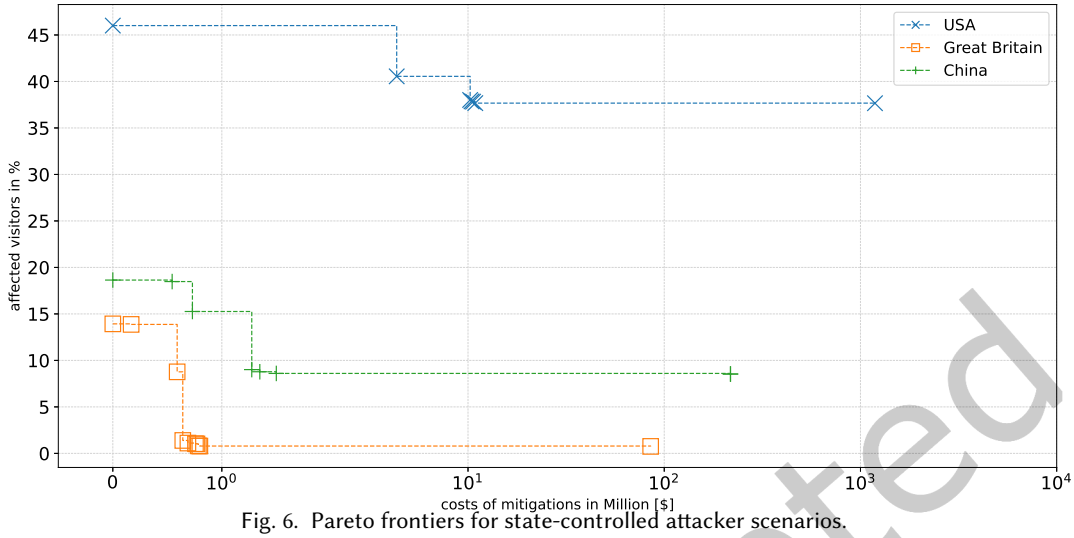


Fig. 6. Pareto frontiers for state-controlled attacker scenarios.

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.

7.1 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 expensive, mitigation against China and GB.

7.2 Performance

The graph-based analysis algorithm discussed in § 6 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

Table 5. 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

graph generation' in Table 5) 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.

8 RELATED WORK

The vulnerability of the internet at the infrastructure level has been studied before [33, 60], including the European BGP topology [53], and web attacks [39], but the analysis of mitigations has been largely ignored. An exception is the analysis of the email infrastructure by Speicher et.al. [42]. They compare mitigations in the email setting and consider countries as defenders and attackers. A major difference is that most email communication is captured by considering all pairs from a small number of providers, which results in a drastically smaller problem size.

Our analysis follows the Stackelberg planning methodology [43], which was originally proposed for mitigation analysis in simulated pentesting. This discipline is closely related to attack graphs, which were first introduced by Philipps and Swiler [72]. Like planning, attack graphs describe an attack (a plan) as a combination of atomic components (actions). Both aim at understanding threats that arise as combinations of atomic actions. There are many flavors of attack graphs, including the *monotonic* formulation, where only positive preconditions and postconditions are permitted [26, 44, 51, 52, 58, 59, 82]. The attacker task in our case is monotonic as well, it keeps gaining new assets, but never lose any assets during the attack. Likewise, our restriction to attacker rules with singleton postconditions and the requirement of the dependency graph to be acyclic bears resemblance to logical programming and stratified semantics for Datalog programs.

In terms of formal mitigation analysis, our setting relates to game-theoretic security models, specifically to Stackelberg competitions, where the game consists of a single exchange of move and countermove. In our setting,

each 'move' here consists of an entire (defender- respectively attacker-) action strategy. These have been studied to allocate physical defenses (*e.g.,* [80]), deploy air marshals in planes, place honeypots and security resources in network of computers and IoT devices [12, 30–32, 65], and deactivate products or patch vulnerabilities in an enterprise network [21]. In particular, Serra et al. [21] employed a Stackelberg game to compute the trade-off between the impact of vulnerabilities and productivity in an enterprise network. They evaluated per-host protection on synthetic enterprise graphs with up to 30k edges. In contrast, we focus on the analysis of Pareto-optimal defenses on the entire Internet by focusing on global protections. Thus, we have a different trade-off between scalability and precision. We evaluated our algorithm on a snapshot of the Internet based on the Top 5k Alexa domains and attack graphs with more than 2M edges.

Algorithmically, probabilistic defenses against an attacker with uncertainty raises the complexity of the attacker plan task considerably, which is not necessary for our use case: none of the mitigations rely on the adversary's uncertainty about its placement. Another line of research considers graphical security models that include defending nodes (*e.g.,* [14, 15]), so-called attack-defense trees, but scale worse than Stackelberg planning [40].

9 CONCLUSION

We proposed a holistic approach to securing the users from Web-based attacks, based on an extensive model of attacks and defenses (with associated costs and security benefits), and an optimized algorithm based on Stackelberg planning. 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 [19]. Likewise, new technological proposals to improve web security can immediately be added to the mitigation model to compete against existing technologies.

While our approach scales well in the size of the property graph, it does not scale well with the number of possible mitigations, limiting our analysis scenarios where mitigations are adopted globally, instead of host-by-host. Effective pruning techniques on the defender-level are therefore necessary. Some of those presented in [43] and [85] are amenable to our graph exploration approach, but some cannot be expressed with Neo4j's query language. Moreover, heuristics can help to find effective bounds quickly, but have not yet been explored for defender-level planning. Finally, other approaches like constraint optimization, lifted planning [8], or custom approaches for countermeasure selection in attack graphs [40] are worth exploring. A crucial requirement is the ability to read problem descriptions in the size of hundreds of thousands of facts (i.e., the size of the property graph), which research prototypes are often not adapted to. With the present work, we raise the bar for the scalability considerably, but we are confident that further improvements in scalability and flexibility are possible.

REFERENCES

[1] Keumars Afifi-Sabet. 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.
[2] Anonymous. Towards a comprehensive picture of the great firewall's DNS censorship. In *Proc. 4th FOCI*, 2014.
[3] APNIC. Dnssec validation rate by country. Accessed: 01/09/2021 URL: https://stats.labs.apnic.net/dnssec.
[4] Adam Barth and Brandon Sterne. Content security policy 1.0. Technical report, W3C, 2015. URL: http://www.w3.org/TR/2015/NOTE-CSP1-20150219/.

[5] Chromium Blog. A safer default for navigation: Https, 2021. URL: https://blog.chromium.org/2021/03/a-safer-default-for-navigation-https.html.
[6] Mozilla Security Blog. Firefox 91 introduces https by default in private browsing, 2021.
[7] Mike Chapple. How expensive are ipsec vpn setup costs?, 2017. Accessed: 01/06/2021. URL: http://searchsecurity.techtarget.com/answer/How-expensive-are-IPsec-VPN-setup-costs.
[8] Augusto B. Corrêa, Florian Pommerening, Malte Helmert, and Guillem Francès. Lifted Successor Generation Using Query Optimization Techniques. *Proceedings of the International Conference on Automated Planning and Scheduling*, 30:80–89, June 2020.
[9] Alexander Dax and Robert Künnemann. On the soundness of infrastructure adversaries. In *Proc. 34th IEEE CSF*, 2021.
[10] 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.
[11] 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.
[12] Antonino Rullo et al. Pareto optimal security resource allocation for internet of things. *ACM Trans. Priv. Secur.*, 20, 2017.
[13] B. Laurie et al. Certificate Transparency. RFC 6962 (Experimental), June 2013.
[14] Barbara Kordy et al. Foundations of attack-defense trees. In *Formal Aspects in Security and Trust*, 2010.
[15] Barbara Kordy et al. ADTool: security analysis with attack-defense trees. In *Quantitative Evaluation of Systems*, 2013.
[16] Bertil Chapuis et al. An empirical study of the use of integrity verification mechanisms for web subresources. In *Proc. 29th WWW*, 2020.
[17] Bill Marczak et al. An analysis of china's "great cannon". In *Proc. 5th FOCI*, 2015.
[18] D. Raumer et al. Efficient serving of vpn endpoints on cots server hardware. In *Cloud Networking*, 2016.
[19] David Adrian et al. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In *Proc. 22th ACM CCS*, 2015.
[20] David Silver et al. Password managers: Attacks and defenses. In *Proc. 24th USENIX Security*, 2014.
[21] Edoardo Serra et al. Pareto-optimal adversarial defense of enterprise systems. *ACM Trans. Inf. Syst. Secur.*, 17, 2015.
[22] Enrico Gregori et al. On the incompleteness of the as-level graph: a novel methodology for BGP route collector placement. In *Proc. 12th IMC*, 2012.
[23] Eric Osterweil et al. Reducing the x. 509 attack surface with dnssec's dane. *Securing and Trusting Internet Names, SATIN*, 12, 2012.
[24] Geng Hong et al. How you get shot in the back: A systematical study about cryptojacking in the real world. In *Proc. 25th ACM CCS*, 2018.
[25] Hyeonmin Lee et al. A longitudinal and comprehensive study of the DANE ecosystem in email. In *Proc. 29th USENIX Security*, 2020.
[26] Hyunyoung Kil et al. Graph theoretic topological analysis of web service networks. *World Wide Web*, 12, 2009.
[27] J. Hodges et al. HTTP Strict Transport Security (HSTS). RFC 6797 (Proposed Standard), November 2012.
[28] Jennifer Rexford et al. BGP routing stability of popular destinations. In *Proc. 2th ACM IMW*, 2002.
[29] Joel Weinberger et al. Subresource integrity. Technical report, W3C, 2016. URL: http://www.w3.org/TR/2016/REC-SRI-20160623/.
[30] Karel Durkota et al. Approximate solutions for attack graph games with imperfect information. In *Proc. 6th GameSec*, 2015.
[31] Karel Durkota et al. Optimal network security hardening using attack graph games. In *Proc. 24th IJCAI Conference*, 2015.
[32] Karel Durkota et al. Case studies of network defense with attack graph games. *IEEE Intelligent Systems*, 31, 2016.
[33] Kevin R. B. Butler et al. A survey of BGP security issues and solutions. *Proc. IEEE*, 98, 2010.
[34] Lukas Weichselbaum et al. CSP is dead, long live csp! on the insecurity of whitelists and the future of content security policy. In *Proc. 23th ACM CCS*, 2016.
[35] Malik Ghallab et al. *Automated Planning: Theory and Practice*. Morgan Kaufmann, 2004.
[36] Marius Steffens et al. Don't trust the locals: Investigating the prevalence of persistent client-side cross-site scripting in the wild. In *Proc. 26th NDSS Symposium*, 2019.
[37] Marius Steffens et al. Who's hosting the block party? studying third-party blockage of csp and sri. In *Proc. 28th NDSS Symposium*, 2021.
[38] Michael Weissbacher et al. Why is CSP failing? trends and challenges in CSP adoption. In *Proc. of RAID-14*, 2014.
[39] Milivoj Simeonovski et al. Who controls the internet?: Analyzing global threats using property graph traversals. In *Proc. 26th WWW*, 2017.
[40] Orly Stan et al. Heuristic approach for countermeasure selection using attack graphs. In *Proc. 34th IEEE CSF*, 2021.
[41] P. Hallam-Baker et al. DNS Certification Authority Authorization (CAA) Resource Record. RFC 8659, November 2019.
[42] Patrick Speicher et al. Formally reasoning about the cost and efficacy of securing the email infrastructure. In *Proc. 3th IEEE EuroSP*, 2018.
[43] Patrick Speicher et al. Stackelberg planning: Towards effective leader-follower state space search. In *Proc. 32th AAAI*, 2018.
[44] Paul Ammann et al. Scalable, graph-based network vulnerability analysis. In *Proc. 9th ACM CCS*, 2002.
[45] R. Arends et al. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), March 2005.
[46] Radhesh Krishnan Konoth et al. Minesweeper: An in-depth look into drive-by cryptocurrency mining and its defense. In *Proc. 25th ACM CCS*, 2018.
[47] Ricardo V. Oliveira et al. The (in)completeness of the observed internet as-level structure. *IEEE/ACM Trans. Netw.*, 18, 2010.
[48] Sebastian Roth et al. 12 angry developers - A qualitative study on developers' struggles with CSP. In *Proc. 28th ACM CCS*, 2021.

[49] Sid Stamm et al. Reining in the web with content security policy. In *Proc. 19th WWW*, 2010.
[50] Stefano Calzavara et al. Content security problems?: Evaluating the effectiveness of content security policy in the wild. In *Proc. 23th ACM CCS*, 2016.
[51] Steven Noel et al. Advances in topological vulnerability analysis. In *Cybersecurity Applications & Technology Conference for Homeland Security*, 2009.
[52] Sushil Jajodia et al. Topological analysis of network attack vulnerability. In *Managing Cyber Threats: Issues, Approaches and Challenges*. Springer, 2005.
[53] Sylvain Frey et al. It bends but would it break? topological analysis of BGP infrastructures in europe. In *Proc. 1th IEEE EuroSP*, 2016.
[54] Taejoong Chung et al. A longitudinal, end-to-end view of the DNSSEC ecosystem. In *Proc. 26th USENIX Security*, 2017.
[55] Tobias Urban et al. Beyond the front page: Measuring third party dynamics in the field. In *Proc. 29th WWW*, 2020.
[56] Victor Le Pochat et al. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proc. 26th NDSS Symposium*, 2019.
[57] Waqar Aqeel et al. On landing and internal web pages: The strange case of jekyll and hyde in web performance measurement. In *Proc. 20th IMC*, 2020.
[58] Xinming Ou et al. A scalable approach to attack graph generation. In *Proc. 13th ACM CCS*, 2006.
[59] Nirnay Ghosh and S. K. Ghosh. An intelligent technique for generating minimal attack graph. In *Proc. Workshop on Intelligent Security*, 2009.
[60] Sharon Goldberg. Why is it taking so long to secure internet routing? *Commun. ACM*, 57, 2014.
[61] P. Hoffman and J. Schlyter. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698 (Proposed Standard), August 2012.
[62] R. Housley. Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP). RFC 4309 (Proposed Standard), December 2005.
[63] ICANN. Tld dnssec report, 2021. Accessed: 01/08/2021. URL: http://stats.research.icann.org/dns/tld_report/.
[64] Joshua Mervine Justin Dorfman. 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.
[65] Karel Durkota et al. Game-theoretic algorithms for optimal network security hardening using attack graphs. In *Proc. 14th AAMAS Conference*, 2015.
[66] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), December 2005.
[67] MaxMind. IP Geolocation and Online Fraud Prevention. http://dev.maxmind.com/, 2017.
[68] MDN. Certificate transparency, 2021. Accessed: 23/07/2021. URL: https://developer.mozilla.org/en-US/docs/Web/Security/Certificate_Transparency.
[69] European Network and Information Security Agency. Study of the cost of dnssec deployment, 2009. URL: https://www.enisa.europa.eu/publications/archive/dnsseccosts.
[70] Sean Oesch and Scott Ruoti. That was then, this is now: A security evaluation of password generation, storage, and autofill in browser-based password managers. In *Proc. 29th USENIX Security*, 2020.
[71] Cynthia Phillips and Laura Painton Swiler. A graph-based system for network-vulnerability analysis. In *Proceedings of the 1998 Workshop on New Security Paradigms*, NSPW '98, pages 71–79, New York, NY, USA, January 1998. Association for Computing Machinery.
[72] Cynthia Phillips and Laura Painton Swiler. A graph-based system for network-vulnerability analysis. In *New Security Paradigms Workshop*, 1998.
[73] Python. tldextract 3.1.0, 2020. URL: https://pypi.org/project/tldextract/.
[74] E. Rescorla. HTTP Over TLS. RFC 2818 (Informational), May 2000.
[75] RIPE Atlas. Internet data collection system. https://atlas.ripe.net/, 2018.
[76] RIPE Stat. Information about specific IP addresses and prefixes, 2017. URL: https://stat.ripe.net/.
[77] Bruce Schneier. Attack trees. *Dr. Dobb's journal*, 24(12):21–29, 1999.
[78] Statcounter. Desktop browser market share worldwide. Accessed: 23-07-2021. URL: http://gs.statcounter.com/browser-market-share/desktop/worldwide.
[79] Ben Stock and Martin Johns. Protecting users against xss-based password manager abuse. In *Proc. of 9th ASIACCS*, 2014.
[80] Milind Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2011.
[81] MyEtherWallet #teamMEW. 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.
[82] Steven J. Templeton and Karl E. Levitt. A requires/provides model for computer attacks. In *New Security Paradigms Workshop*, 2000.
[83] Giorgio Di Tizio. Drive-by download attacks as a stackelberg planning problem. Master's thesis, University of Trento, 2018.
[84] Ben Toews. Subresource integrity, 2015. URL: https://githubengineering.com/subresource-integrity/.
[85] Álvaro Torralba et al. Faster Stackelberg Planning via Symbolic Search and Information Sharing. *Proc. of the 35th AAAI Conference on Artificial Intelligence*, 2021.

Table 6. Threat Model Predicates

Predicate	Description
$x \xrightarrow{\text{loc}} cn$	$x \in AS \cup IP \cup D \cup NS$ is located in $cn \in Country$
$d \xrightarrow{\text{A}} i$	$d \in D \cup NS$ has address $i \in IP$
$i \xrightarrow{\text{orig}} a$	$i \in IP$ belongs to $a \in AS$
$c \xrightarrow{\text{JS}} d$	$d \in D$ 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
$e \xrightarrow{\text{DNS}} d$	$e \in NS$ is one of the authoritative name servers of d
$p \xrightarrow{parent\_zone} e$	$p \in NS$ 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
$C(x)$	$x \in AS \cup IP \cup D \cup Country \cup NS \cup CA$ is compromised. In case $x \in D \cup NS$, x can be used to directly (indirectly) affect user's visits (*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 (*Website access compromised*).
$C^{\text{web}}(c, d)$	The website on $d \in D$ is considered compromised for all the visitors from $c \in Country$. (*Website access compromised from c*)
$XSS(d)$	$d \in D$ is vulnerable to XSS
$Upgrade\ Requests(d)$	$d \in D$ employs the field `upgrade-insecure-requests` in the CSP to force HTTPS for all the resource requests.
$SRI(d, c)$	$d \in D$ 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
$HSTS(d)$	$d \in D$ implements the header `strict-transport-security`
$Redirect(d)$	$d \in D$ redirects HTTP connections to HTTPS. The redirection is either temporary or permanent
$CT(d)$	The digital certificates, for $d \in D$, are signed by CAs that are compliant with the CT
$DANE(d)$	$d \in NS$ implements DANE
$I^{\text{DNS}}(d)$	The DNS resolution of $d \in D$ is compromised (*Globally compromised DNS*).
$I^{\text{DNS}}(d, e)$	The DNS resolution of $d \in D$ is compromised for the visitors from $c \in Country$ (*Country compromised DNS*)
$I^{\text{R}}(i, j)$	The route between $i, j \in IP$ is compromised
$I^{\text{CA}}(d)$	$d \in D$ is vulnerable to certificate authority attacks
$TLSA\_0(a)$, $TLSA\_2(a)$	$a \in CA$ is present in the certificate chain of the TLSA record with certificate usage field 0 or 2

A FORMAL MODEL OF THE ATTACKER

This appendix contains the complete threat model used to describe Web-based attacks. Table 6 contains the entire list of predicates used in the model. 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*.

Using the notation of the Boolean Logic, the symbol ¬ in front of a predicate negates the predicate itself.

A.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.

A.1.1 Initially Compromised Nodes.

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

`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.

A.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 CSP 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 [37].

A.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[14]. An attacker can modify the DNS resolution and map the domain name to a different IP.

[14]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.

A.1.4 Route Compromise.

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

`Intuition:` If a route from one AS to another is IPsec protected 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. Note that the case in which an endpoint is compromised is captured by (2), which would mark the respective IP and thus domain or name server compromised.

A.1.5 Route to Web Server Compromise.

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

`Intuition:` If a route between a client and a web server is insecure, assuming the worst scenario in which a *non-tech-savvy* user accesses the web server via HTTP (at the time of writing HTTP is the default protocol used by browsers if a protocol is not explicitly defined), 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 [27];
- *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 it is vulnerable to certificate authority attacks[15], then a malicious CA can forge digital certificates for the domain and use them to authenticate connections to malicious web servers.

In *Case 3* we ignore the use of a permanent redirection and we require, in addition, the presence of the `strict--transport-security` header. This choice is due to the fact that the redirection is not a secure mitigation and the HSTS provides a better security with respect to the Permanent redirection:

- HSTS covers the entire domain;
- HSTS implements a preloaded list[16];

[15]See rules: 16, 17, 18

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

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.[17] For a first approximation, we assumed the attacker to not be allowed to exploit this vulnerable window.
The *postcondition* declares that all the connections originated from the country where the sender AS is located, are compromised. This is an upper bound assumption because there could exist ASes 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.

A.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. Thus, the DNS resolution of the domain is compromised for all connections originating in the country where the client AS is located[18].

A.1.7 From DNS to Domain Compromise.

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

`Intuition:` If a web server has a *Globally compromised DNS*[19] and either it does not fulfill all the conditions to establish a secure connection or the attacker is able to forge a malicious certificate for the website, then the attacker can redirect **all** the clients to a malicious web server that can claim to be the legitimate one.

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

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

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

`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** these conditions are met:

- *The web server does not implement SRI*, thus the JS resource can be replaced with a malicious one.
- *The protocol to retrieve the resource from c is not HTTPS or the attacker is able to forge a certificate for the CDN.*

[17] A possible mitigation for this scenario is to increase the number of domains contained in the preloaded HSTS list.
[18] This is the same simplification presented in rule 9
[19] This means that the attacker has control over the content provided by one of the authoritative NSs for this domain.

- *The web server does not implement the* `upgrade-insecure-requests` *field in the CSP or the attacker is able to forge a 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 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 *post condition* presents the same structure of rule 12.

A.1.8 Inline JS Injection.

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

`Intuition:` If the route from a client to a CDN, that provides JS resources to a web server, is insecure[20] and **all** these conditions are met:

- *The web server does not implement SRI:* 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 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 certificate for the CDN*

Then, the attacker can intercept the JS requests and inject malicious JS code. Note that we required that any mitigation (*UpgradeRequests* or *l_HTTPS*) does not break the functionality of the website by requiring that the resource is available over HTTPS (see §A.2).

A.1.9 Certificate Compromise.

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

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

- *The web server's digital certificates are signed by CAs that are not compliant with the Certificate Transparency project.*

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

- *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.

$$\frac{\begin{array}{cccc} a{\in}CA & d{\in}D & e{\in}NS & C(a) \\ e \xrightarrow{DNS} d & \neg CT(d) & C(e) & \end{array}}{I^{\text{CA}}(d)} \tag{17}$$

`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.

$$\frac{\begin{array}{cccccc} a{\in}CA & d{\in}D & e{\in}NS & C(a) & e \xrightarrow{DNS} d & \\ \neg C(e) & (TLSA\_0(d,a) \vee TLSA\_2(d,a)) & & & & \end{array}}{I^{\text{CA}}(d)} \tag{18}$$

`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*[21]
- *The TLSA certificate usage field is 2 and the compromised CA is in the Certificate Chain from the Server certificate to the Trust anchor.*

A.1.10 Third-party JS Injection.

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

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

A.1.11 Access compromised to website access compromised.

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

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

A.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.

A.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.

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

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

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

A.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 (due to mixed-content). 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 the 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 presence of HTTPS on the domain.

A.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.

A.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.

A.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.

B GENERIC ATTACK GRAPH GENERATION ALGORITHM

Algorithm 2: property graph to attack graph (generic)

Input: property graph PG, dependency graph DG, initial assets attacker
Output: attack graph AG
`// initialize AG`
AG $\leftarrow (V \cup \{\texttt{attacker}\}, \{(\texttt{attacker}, v) \mid v \in V\})$ with $V = \{C(x) \mid x \in \textit{initial assets attacker}\}$;
$(\text{DG}', \gamma) \leftarrow$ subst-cycles(DG) `// returns graph with dummy nodes instead of cycles and a mapping` γ `from those dummy nodes to the subgraph they substituted.`
$\vec{r} \leftarrow$ topological-sorting(DG′)
for r_i *in* $\vec{r}$ *(in order)* **do**
  if $r_i \in \text{dom}(\gamma)$ **then**
    while *fixpoint not reached* **do**
      for r'_j *in* topological-sorting($\gamma(r_j)$) *(in order)* **do**
        AG $\leftarrow$ AG + $\{\sigma(v) \rightarrow \sigma(w) \mid \text{PG}, \sigma \vdash \text{graph}(r'_j) \wedge v \in \textit{pre}(r'_j) \wedge w \in \text{post}(r'_j)\}$;
  else
    AG $\leftarrow$ AG + $\{\sigma(v) \rightarrow \sigma(w) \mid \text{PG}, \sigma \vdash \text{graph}(r_i) \wedge v \in \textit{pre}(r_i) \wedge w \in \text{post}(r_i)\}$;
`// add mitigations. (Note that defender rules contain mitigation disabling dependencies)`
mark all edges in AG as removable if a defender rule applies