Who Controls the Internet? Analyzing Global Threats using Property Graph Traversals

Milivoj Simeonovski
CISPA, Saarland University
Saarland Informatics Campus
simeonovski@cs.uni-saarland.de

Giancarlo Pellegrino
CISPA, Saarland University
Saarland Informatics Campus
gpellegrino@cispa.saarland

Christian Rossow
CISPA, Saarland University
Saarland Informatics Campus
rossow@cispa.saarland

Michael Backes
CISPA, Saarland University &
MPI-SWS
Saarland Informatics Campus
backes@cs.uni-saarland.de

ABSTRACT

The Internet is built on top of intertwined network services, e.g., email, DNS, and content distribution networks operated by private or governmental organizations. Recent events have shown that these organizations may, knowingly or unknowingly, be part of global-scale security incidents including state-sponsored mass surveillance programs and large-scale DDoS attacks. For example, in March 2015 the Great Cannon attack has shown that an Internet service provider can weaponize millions of Web browsers and turn them into DDoS bots by injecting malicious JavaScript code into transiting TCP connections.

While attack techniques and root cause vulnerabilities are routinely studied, we still lack models and algorithms to study the intricate dependencies between services and providers, reason on their abuse, and assess the attack impact. To close this gap, we present a technique that models services, providers, and dependencies as a property graph. Moreover, we present a taint-style propagation-based technique to query the model, and present an evaluation of our framework on the top 100k Alexa domains.

Keywords

Cyber-attacks; (DoS) denial of service attacks; property graph traversals

1. INTRODUCTION

About half of the world population is using the Internet every day to communicate with friends, read newspapers, and carry out financial transactions. These services rely on core operations such as IP routing, domain name resolution, and email transfers, which are carried out by organizations ranging from universities and governmental agencies to private-sector organizations. Such organizations thus have extensive power, which, if misused, can result in global-scale security violations. Service providers can perform various attacks such as advertising false BGP paths to sensitive targets through their network [3, 4] and injecting HTTP responses into TCP connections [17]. Even more severe security violations can be performed when providers cooperate. Recent events have shown that cooperation between providers and state authorities resulted in global-scale security incidents such as mass surveillance, e.g., the PRISM program [14], and distributed denial-of-service (DDoS) attacks, e.g., the Great Cannon attack [19, 16].

Service providers can also be victims of attacks. The Internet is often considered as a model of resilience due to its distributed and decentralized design. While this applies in cases of random node failures, it does not guarantee survivability of the network with targeted attacks [5, 26]. For example, attackers can focus their efforts against a few, carefully selected providers to disrupt network operations at large scale. These types of attacks have already been observed against root name servers, the servers at the top of the DNS hierarchy, so far they have had very limited impact. However, the skills and power of attackers are increasing, and the DNS infrastructure of Dyn.com, which serves popular websites, was struck by two DDoS attacks. While the volume of the attacks has not yet been disclosed, this attack caused outages in the name resolution of popular services such as Amazon, Netflix, Twitter, Reddit, and Spotify.

An increasing number of reports and studies are showing that a limited number of players have an important influence on the overall security of the Internet infrastructure. While we have a good understanding of attack techniques [16, 17], attackers [24], and victims [21], we have a rather limited capability to assess the impact of attacks against, or performed by, core service providers. In the past decades, the security of the Internet core infrastructures has been under continuous scrutiny. Many works focused on different facets, using analysis techniques such as topological analyses (e.g., [12, 13, 10]) and traditional threat analysis via attack enumeration (e.g., [6]). However, the contribution of these works is limited to a single core service, and, to date, the interdependencies between core services remain largely unexplored.

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3038912.3052587

In this paper, we take a step forward and propose an investigation technique to assess global-scale threats. We present a model of the Internet infrastructures based on *property graphs.* Nodes of our model are servers, organizations, and autonomous systems, that are connected with edges to represent relationships. To mine data from our model, we present a combination of taint-style techniques and propagation rules, which is automatically translated in graph traversals. We assessed our approach on a model with 1.8 millions of nodes and 4.7 millions of relationships. Starting from the top 100K Alexa domains, we built our model using publicly available resources, e.g., RIPE Atlas, and by acquiring relationships via Web service crawling. Finally, we mined our model to assess the impact of attacks. We present six metrics to select attacker and victim candidates. Then, we measure the impact of three different attack scenarios, which are based on the Great Cannon attack, the PRISM program, and the DDoS against Dyn.com.

Our results show that already just a few players may have an extensive power: 14 countries and 14 autonomous systems can, directly or indirectly, affect the security of about 23% of websites. Our analysis show that the United States is the country with the largest fraction of power, i.e., 16% of websites, and that network operators, albeit of moderate size such as Google[1], can match in terms of affected websites the aggregate result of large countries like Russia, Germany, Japan, and China. In addition, our results show that little has been learned from past attacks. For example, 70% of JavaScript (JS) inclusion is still done over unprotected connections, i.e., via HTTP URLs, which can be used to mount the Great Cannon attack. Finally, our results indicate that the DDoS attack against Dyn.com was the result of careful choice. Dyn's ASes host authoritative name servers used directly and indirectly by 3% to 5% of the 100K Alexa domains.

This paper provides the following contribution:

- We present a first study on attacks based on dependencies between Internet core services;
- We present a framework to model and reason on global-scale threats;
- We present a taint-style technique based on propagation rules and property graphs to quantify the impact of security incidents;
- We assess our technique on 1.8M data items acquired from the top 100K Alexa domains.

2. BACKGROUND

Before presenting our framework, we describe relevant case studies and introduce our threat model.

2.1 Case Studies

Our study is motivated by three recent, large-scale and well-known security incidents.

The Great Cannon DDoS Attack—On March 16th, 2015, `Greatfire.org`, a non-profit organization monitoring Internet censorship in China, and GitHub, the hosting provider, were victim of a large DDoS attack, among the largest DDoS ever experienced by GitHub [19]. The attack was caused by malicious JavaScript code, which was injected into TCP connections crossing the Chinese network borders [16, 11]. The injected code turned Web browsers into an HTTP-based DDoS botnet by aggressively requesting resources from the targets [16, 11, 22].

The PRISM Program—On June 7th 2013, *The Guardian* documented PRISM, a National Security Agency surveillance program with direct access to Internet communications and stored information including emails, chats, and VoIP calls, from servers of popular tech companies such as Microsoft, Yahoo, Google, and Facebook [14]. While the direct involvement of popular tech providers is still unclear, in this paper, we make the assumption that establishing this type of collaboration is possible and can be voluntary, or coerced by authorities by means of law and court orders.

The DDoS Attack Against Dyn.com—On October 21st 2016, the DNS infrastructure of Dyn.com was struck with two DDoS attacks. According to Dyn.com,[2] the attack caused increased DNS query latency and delayed zone propagation. As a result, Dyn.com customers, including Amazon, Netflix, Twitter, Reddit, and Spotify, experienced outages on the name resolution. At the time of writing, the details of the attacks are not published, however, the outage clearly affected hundreds of millions of Internet users, who could not access the online services of Netflix and Twitter.

2.2 Threat Model

From our three case studies, we derive the threat model for this paper. We focus on attacks that can target and involve a large number of individuals and organizations around the globe. We represent attacks as a set of three elements: an attacker, the attack goal, and the attack technique.

Attacker—Attackers can be a service provider, a group of providers, or a country. In case the attacker is the provider, we consider the attackers: domain name provider, email provider, network provider, content distribution network, and domain name owners.

Service cooperation can be achieved via collaboration between two or more attackers, or via a centralized coordinator, e.g., state-sponsored attacks. In both cases, we assume that colluding attackers have a shared, collective memory and information acquired by one attacker is available to other attacker.

Goal—The attack goal answers the question *what* attackers intend to achieve. From our case studies, we consider three goals: DDoS via distribution of malicious JavaScript, acquisition of emails, and DoS against service providers.

Technique—To achieve their goals, attackers can use different techniques. For example, law enforcement agencies may require access to user's email boxes. Network providers may intercept TCP traffic traversing their own autonomous system (AS) to inject malicious JavaScript code. In this paper, we consider the following techniques: email sniffing, redirection via malicious domain resolution, in-path content injection, and hosting malicious content.

3. MODELING FRAMEWORK

We now present our modeling framework. We base our model on *labeled property graphs.* Labeled property graphs store information in nodes and edges in the form of key-value *properties.* We present property graphs in details in Section 3.1. We represent elements such as domain names,

[1] Google is also a network operator as it controls the autonomous system 15169

[2] `http://dyn.com/blog/dyn-statement-on-10212016-ddos-attack/`

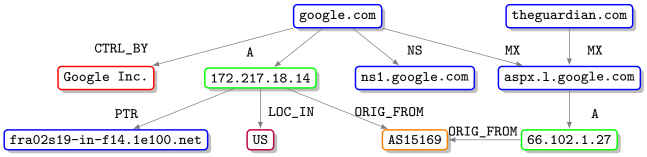

Figure 1: Excerpt of property graph for `google.com`

IPs, organizations, and countries as nodes. Then, we use edges to represent relationships between nodes. For example, if a domain name resolves to an IP, then we add an edge between the two nodes. In a similar way, we represent relationships between IPs and countries in the sense that if an IP is located in a country, then we place an edge between the country and the IP. Finally, we use labels to specify the type of relationship.

We mine information from graphs using *graph queries*. Graph queries allow to visit graphs based on nodes, edges, and properties. In this paper, we used a technique based on taint-style propagation technique and propagation rules. Starting from an initial set of nodes, we propagate a taint value according to a list of rules. Propagation rules are presented in Section 3.2, and queries and their evaluation are presented in Section 3.3.

3.1 Property Graph

A labeled property graph $G = (V, E, \lambda, \mu)$ is a directed multigraph where V is a set of nodes, $E \subseteq (V \times V)$ is a set of edges, $\lambda : V \cup E \rightarrow \Sigma$ is a function that labels nodes and edges with symbols of the alphabet Σ, and $\mu : (V \cup E) \times K \rightarrow S$ is a function that associates key-value properties, e.g., (k, s) where $k \in K$ is the key and $s \in S$ is the string value, to nodes and edges.

Figure 1 shows an excerpt of a property graph. We use node labels to type model elements. For example, we use the label *Domain* for Internet domain names, e.g., `google.com` and *Address* for IP addresses, e.g., `172.217.18.14`. We use node properties to store element data. For example, we use the key *IPv4* for nodes *Address* to store the dot-decimal notation for IPv4 addresses. Our model uses other types of nodes including *Organization*, *Autonomous System* (AS), and *Country*. The full list of node labels and properties is shown in Table 1.

When we can establish a relationship between elements, we place an edge, with the label specifying the type of relationship. Relationships can be established, for example, with DNS queries and publicly available databases such as RIPE Atlas. We present data acquisition in detail in Section 4. With reference to Figure 1, the domain name `google.com` resolves to `172.217.18.14` (DNS record type A), which is hosted in the AS number 15169 and geolocated in the United States. These three relationships are represented with edges labeled with `A`, `ORIGIN_FROM`, and `LOC_IN` respectively. Then, `google.com` has four authoritative DNS server one of which is `ns1.google.com`. The domain name `google.com` has also an email server `aspmx.l.google.com` whose IPv4 is `66.102.1.27`, also hosted in AS 15169. When we can also establish ownership of elements such as domain names, then we place edges between the *Organization* and the element. For example, in Figure 1 we have a node `Google Inc.` which is the organization that owns the domain `google.com`. We represent this relationship with an edge `CTRL_BY`. Finally, Figure 1 shows a relationship that exists between the domain `theguardian.com` and the email server `aspx.l.google.com`. The complete list of edge types is shown in Table 1.

Labels	Description
Address	Node for IP address
Domain	Node for a domain name; the source data set, e.g., Alexa or JS, is a node property
DNS Zone	The zone administrated by an authoritative name server
AS	IANA number assigned to the AS; The hosted IPs is a node property
Country	Code Country code, number of IPs
Organization	Service provider name
`ORIG_FROM`	*AS* where an *Address* originates from
`LOC_IN`	*Country* where an element is located
`CTRL_BY`	*Organization* controlling, e.g., a *Domain*
`A`	DNS record mapping *Address* to *Domain*
`MX`	DNS record mapping *Domain* for email delivery
`NS`	DNS record for name servers
`ZONE`	DNS record for authoritative information of a *DNS zone*
`CNAME`	Aliases from *Domain* to *Domain*
`PTR`	PTR DNS record type maps an *Address* to a *Domain*
`INCL_JS_FROM`	*Domain* name or *Address* hosting JS library

Table 1: Labels of nodes and relationships

3.2 Taint-style Propagation and Rules

A central concept of our framework is taint-style propagation and propagation rules. These elements are the building blocks to specify queries. The idea behind propagation rules is that each node of the graph may become compromised by an attacker. For example, if an attacker controls a host, then the *Address* node is considered compromised. As a consequence of this fact, *Domain* nodes that resolve to the compromised *Address* are compromised as well. The "propagation" of compromise between nodes follow specific rules that depend on the attack. Attacks may result in less severe consequences for node elements. Consider, for example, the Great Cannon attack. Web sites that included JS hosted in malicious networks can be considered compromised as well. However, in the specific case of the Great Cannon, the malicious JS code did not perform attacks against the originating server. Thus, in this case, no further entities are compromised.

Our framework supports an arbitrary granularity for compromise levels. In this paper, we use three levels with the following symbols: $\mathbf{c} \in \Sigma$ for (completely) compromised, $\mathbf{pc} \in \Sigma$ for partially compromised, and $\perp \in \Sigma$ for non-compromised. When a node n is compromised (i.e., $\mathbf{c}$), we add the compromise level as a node property $\mathcal{C}$, e.g., $\mu(n, \mathcal{C}) = \mathbf{c}$. The propagation is implemented via rules. Each rule is a pair of preconditions and postconditions. Preconditions are evaluated on the graph. If they hold, then postconditions will hold in the graph. This is achieved by modifying the graph such that postconditions will match. The general form of a rule is the following:

$$\frac{pre}{post} \tag{r}$$

Where *pre* and *post* are two predicates for pre and postcondition, respectively.

With reference to the previous example, the propagation rule based on the `A` (name lookup) edge is the following:

$$\frac{\mu(n, \mathcal{C}) = \mathbf{c}, e = (m, n) \in E, \lambda(e) = \mathtt{A}}{\mu(m, \mathcal{C}) := \mathbf{c}} \qquad (r_{\mathtt{A}})$$

This rule can be read as follows: If node n is compromised, there is an edge e between m and n, and the label of e is `A`, then we mark m as compromised.

We use similar rules to Rule $r_{\mathtt{A}}$ for other type of relationships. For example, for `MX` edges we have the following rule:

$$\frac{\mu(n, \mathcal{C}) = \mathbf{c}, e = (m, n) \in E, \lambda(e) = \mathtt{MX}}{\mu(m, \mathcal{C}) := \mathbf{c}} \qquad (r_{\mathtt{MX}})$$

Rules $r_{\mathtt{A}}$ and $r_{\mathtt{MX}}$ can be applied in sequence. For example, let us assume that an address node n is compromised. Then, according to Rule $r_{\mathtt{A}}$, any domain m resolving to n, i.e., $(m, n) \in E$, is also compromised. If m is a domain for mail exchange, according to Rule $r_{\mathtt{MX}}$, any domain p using m as its mail exchange server, i.e., $(p, m) \in E$, is also compromised. In general, starting from a compromised node and a set of rules, we can propagate values $\mathbf{c}$ to other nodes.

Propagation rules are also used to represent weaker forms of compromise. Consider the case in which m is a web server hosting shared JS libraries. If m is compromised, it can, for example, distribute malicious JS libraries, which can be included in third-party websites q. As a result of this, users of q will execute the malicious code. However, this type of compromise may not entirely compromise the server of q, instead it can be used to attack other servers or compromise a user session. We model these forms of compromise with the following rule:

$$\frac{\mu(n, \mathcal{C}) = \mathbf{c}, e = (m, n) \in E, \lambda(e) = \mathtt{JS}}{\mu(m, \mathcal{C}) := \mathbf{pc}} \qquad (r_{\mathtt{JS}})$$

3.3 Query and Evaluation

We can now define more precisely a query to our model and its evaluation. A query $\mathcal{Q} = (I, R, \gamma)$ is composed of three elements: initial set of source nodes I, a set of rules $R = \{r_1, r_2, \cdots, r_n\}$, and result function γ. The set I contains nodes in G, i.e., $I \subseteq N$. For example, if we want to evaluate an attack, source nodes are the initial nodes under control of the attacker and we mark them as compromised. Then, the set R is a set of rules, starting from source nodes, that propagate the taint to other nodes. Finally, the result function γ is a generic function that given the graph G transformed by the propagation rules returns a data value.

The algorithm to evaluate a query $\mathcal{Q}$ is shown in Listing 1. The algorithm is divided into three parts. The first part from Line 3 to Line 7 initializes node labels of G. Each node n in the initial set of compromised nodes I is marked accordingly, i.e., $\mu(n, \mathcal{C}) := \mathbf{c}$. The remaining nodes are initialized with the symbol $\perp$. The second part of the algorithm from Line 10 to Line 16 applies the propagation rules. We use an auxiliary queue Q where we keep the rules to apply. This part of the algorithm loops over the queue until it is empty. At each iteration, we retrieve a rule r from Q, check whether the precondition holds, and apply the post conditions. The resulting graph is stored in G. If the preconditions of r still holds also in the new graph, then we enqueue r in Q. The loop terminates when Q is empty, i.e., all preconditions no longer hold. Finally, we apply the result function and return the result.

Listing 1: Attack Evaluation

```
1  def evaluate($G$, $\mathcal{Q}$)
2    # Node property initialization
3    for $n$ in $N$:
4      if $n$ in $I$:
5        $\mu(n,\mathcal{C})$ := $\mathbf{c}$
6      else:
7        $\mu(n,\mathcal{C})$ := $\perp$
8
9    # Taint Propagation
10   Q := $R$
11   while Q != $\emptyset$:
12     $r$ := Q.pop()
13     if match_pre($r$, $G$) is True:
14        $G$ := apply_post($r$, $G$)
15        if match_pre($r$, $G$) is True:
16          Q.enqueue($r$)
17
18   # Query result
19   return $\gamma(G)$
```

4. DATA SETS AND ACQUISITION

We instantiated our model on a data set of 1.8M nodes from which 350k are unique IP addresses, 1.1M are domain names and 12k are autonomous system. These nodes are connected with 4.7M relationships. Our acquisition starts from popular domains and it is expanded with server and network information. Finally, we add organizations and countries.

4.1 Initial Domain Names

We built our data set starting from domains that individuals and organizations may use for carrying out their daily activities. For this purpose, we used the top 100k Alexa domains, a data set of popular domain names maintained by Alexa.[3] For each domain, we created a node *Domain*. Our model contains additional domains that were implicitly acquired via Web crawling starting from the Alexa domains. To distinguish the origin of a domain name, we use a node property $\mathcal{O}$ that flags a node according to its origin.

4.2 Servers

Starting from the initial domain names, we resolve hosts that are responsible for core operations, i.e., web servers, authoritative name servers, email servers, content distribution servers, and routers. The collection of data is done via Domain Name System queries and a Web crawler.

Authoritative Name Servers—The DNS records of a domain name are maintained by the authoritative name servers. Each authoritative name server is responsible for a portion of the domain name space, the so-called *DNS zone*. DNS zone information is stored in the SOA record type. For each domain, we retrieve the SOA record, and add a node *Zone* connected with an edge `ZONE` to the domain. Then, we retrieve the fully-qualified domain name of authoritative name servers which are listed in the NS records. For each NS record, we add a node *Domain* connected with an edge

[3] `http://www.alexa.com/`

NS to the zone node of the domain. In addition, for each NS domain name, we resolve the IP addresses, and we add a node *Address* with the IP and an edge A from the domain to the IP.

Web Servers—Our initial data set is composed of domains of popular websites. By resolving the domain name, we obtain the IPs of the web server. For each of these IPs, we add a node *Address* in our model and place an edge A between the domain and the address. Domain names may be also have aliases via the DNS CNAME record. In this case, we add the alias domain in the graph and link with a CNAME edge. Then, we further resolve the alias domain and add an *Address* node with A edge.

Email Servers—Next, we identify email transfer agents. When email clients want to send an email to a recipient, they request the MX record of the domain name of the email address. The MX record can be a list of IP addresses and domains. For each IP, we add a node *Address* and connect it with an MX edge to the domain. For each domain, we add a node *Domain* in the graph and the MX edge. Then, we resolve the domain name into an IP address and add a node *Address* with a A edge to the MX domain.

Content Distribution Networks—More and more websites include JS libraries that are hosted on third-party servers. For example, websites can include JS code of advertisement network services to show advertisements to their users. Websites can also use JS frameworks to support website functionalities, e.g., user interface or communication with the server side. Among the popular frameworks we have, for example, jQuery and Angular.js.

Starting from a list of domain names, we identify these JS "include" relationships with a web crawler. We first visit the website and then retrieve all tags to external JS code. We also extract links to internal web pages, e.g., anchor tags, and repeat the analysis on the page of the new links. We repeat this operation for a depth of 2. For each of the retrieved JS URLs, we add an *Address* node if the host is an IP, and a *Domain* node if it is a domain name. For each edge, we store the URL scheme as property using the key $\mathcal{S}$. For example, if the included JS is unprotected, i.e., HTTP, then $\mathcal{S} = \texttt{HTTP}$.

4.3 Routing Information and Networks

We now add information about servers' networks.

Autonomous Systems—An autonomous system is a collection of IP networks and routers which are under the control of a network operator. We retrieve the origin AS of an IP using the RIPEStat database service by RIPE NCC [2]. For each AS, we create an *AS* node and add an edge from the *Address* node to the *AS* node. We additionally retrieve the total number of prefixes announced by an AS and store this number as a node property.

4.4 Countries and Organizations

Finally, we include countries and organization information in the graph. Our goal is to establish a relationship between these entities and the servers of Section 4.2. There are three ways to establish that, i.e., at IP level, at AS level, and at domain level.

The first option is to link organizations and countries to individual IPs. This can be achieved via geolocation. Accordingly, we added geolocation data in our model using the MaxMind database [1]. While this can be achieved for countries, we are not aware of a database or an automated technique to associate a single IP to an organization controlling the server. Given the large number of IPs in our database, establishing this relationship manually is not a feasible task. The second option is to link entities to autonomous systems. This mapping is already available in RIPEstat and we include it in our model. The third is to link entities to domain names. The Domain WHOIS protocol can be used to query information about registered domain names including the domain registrant. Depending on the providing server, the structure and content of the provided information vary. WHOIS data is optimized for readability to humans [9] and thus does not have a consistent document format [15]. While a human can easily use WHOIS to retrieve data items for a single domain, it does not scale to a large volume of domain names. As an alternative source of data, we used the X.509 certificates used for HTTPS. X.509 certificates are primarily used to store servers' public-key and the domain names on which the certificate is valid. Additionally, a X.509 certificate contains the organization name to which the certificate has been issued. We included this information in our database.

5. ENTITY IDENTIFICATION

Before assessing attacks, we use our model to select entities that can be either attack victims or the attackers. The selection criteria are based on metrics that reflect the popularity and the influence of entities. To this end, we defined six metrics divided into *first-* and *second-order* metrics. First-order metrics are *basic* metrics which rank entities according to the number of hosted servers. Second-order metrics combine basic metrics and measure the level of influence that an entity may have on third-party services. The most popular entities of our metrics are shown in Table 2.

5.1 First Order Metrics

We start with four first-order metrics, one for each server of our model, i.e., name servers, web servers, email servers, and JS hosting servers. We calculate these metrics using two sets of queries, one for ASes and the other for countries.

Metric #1 (Hosted Alexa Domains)—The first metric counts the number of Alexa domains hosted by an AS or a country. For an AS a, the first propagation rule is the following:

$$\frac{\mu(n,\mathcal{C}) = \mathbf{c}, e = (m,n) \in E, \lambda(e) = \texttt{ORIG\_FROM}}{\mu(m,\mathcal{C}) := \mathbf{c}} \qquad (r_{\texttt{ORIG}})$$

followed by Rule $r_{\texttt{A}}$. These two rules, starting from the source node a, propagate the taint value to all IP addresses and then to domain names. Domain names can originate from the Alexa database, or can be imported during the acquisition. To filter Alexa domains, we refine Rule $r_{\texttt{A}}$ by adding a check on the node property, i.e., $\mu(n,\mathcal{O}) = \texttt{Alexa}$:

$$\frac{\mu(n,\mathcal{C}) = \mathbf{c}, e = (m,n) \in E, \lambda(e) = \texttt{A}, \mu(n,\mathcal{O}) = \texttt{Alexa}}{\mu(m,\mathcal{C}) := \mathbf{c}} \qquad (r_{\texttt{A,Alexa}})$$

Finally, we define a function γ which returns the number of compromised domains.

Metric 1		Metric 2		Metric 3		Metric 4		Metric 5		Metric 6	
Country	**Dom.**	**Country**	**JS**	**Country**	**MX**	**Country**	**NS**	**Country**	**JS/NS**	**Country**	**MX/NS**
United States	30,582	United States	47,910	United States	41,434	United States	34,235	United States	41,231	United States	28,800
Netherlands	4,296	Germany	7,830	Germany	12,047	Germany	6,697	Germany	3,101	Netherlands	14,213
Germany	4,178	China	7,273	Great Britain	6,811	France	3,865	Netherlands	3,045	Ireland	11,440
China	4,158	Netherlands	6,963	France	6,261	Great Britain	3,139	China	3,009	Germany	5,380
Japan	3,053	Great Britain	4,455	Netherlands	6,091	Netherlands	3,116	Russia	2,254	Great Britain	3,116
France	2,526	Japan	4,205	Japan	4,314	Canada	2,244	France	2,084	France	2,996
Great Britain	2,400	France	4,048	Russia	3,923	Russia	2,167	Japan	2,000	Russia	2,588
Russia	1,678	Russia	2,865	Italy	3,293	Turkey	2,143	EU	1,636	Japan	1,663
Canada	1,186	Ireland	1,919	Canada	3,042	Japan	2,126	Great Britain	1,364	Spain	1,421
India	1,087	EU	1,581	Ireland	2,897	Spain	1,662	Spain	1,219	Iran	1,123
Ireland	986	Canada	1,347	Spain	2,703	China	1,617	Canada	801	Canada	933
EU	950	Italy	1,159	Turkey	2,094	Iran	1,552	Singapore	787	China	842
Spain	848	Spain	952	Iran	1,946	Brazil	1,070	Poland	540	Italy	798
South Korea	755	Poland	943	India	1,892	India	954	Iran	474	Turkey	797
ASN Name	**Dom.**	**ASN Name**	**JS**	**ASN Name**	**MX**	**ASN Name**	**NS**	**ASN Name**	**JS/NS**	**ASN Name**	**MX/NS|**
13335 CloudFlare	7,170	16509 Amazon-1	10,085	8075 Microsoft	8,503	16276 Ovh	2,415	16509 Amazon-1	15,429	8075 Microsoft	11,596
16509 Amazon-1	2,816	13335 CloudFlare	5,489	16276 Ovh	2,669	24940 Hetzner	2,131	13335 CloudFlare	4,933	13335 CloudFlare	6,790
14618 Amazon-2	1,892	20940 Akamai	3,004	24940 Hetzner	2,497	16509 Amazon	1,907	33517 DynDNS	3,570	16509 Amazon-1	2,018
20940 Akamai	1,830	14618 Amazon-2	2,207	46606 Unified L.	1,353	46606 Unified L.	1,524	4837 China169	2,008	16276 Ohn	1,969
16276 Ovh	1,025	16276 Ovh	1,970	36351 Softlayer	865	36351 Softlayer	1,345	26496 GoDaddy	1,938	26496 GoDaddy	1,750
37963 Alibaba	779	24940 Hetzner	1,508	26496 GoDaddy	799	32475 SingleHop	1,155	4812 China Tlc.	1,875	24940 Hetzner	1,708
24940 Hetzner	725	15133 EdgeCast	1,360	16509 Amazon-1	643	13335 CloudFlare	699	16552 Tiggee	1,467	33517 DynDNS	1,523
15169 Google	525	37963 Alibaba	940	60781 Leaseweb	579	32244 Liquid Web	674	16276 Ovh	1,307	36351 Softlayer	575
36351 Softlayer	518	36351 Softlayer	910	15169 Google	568	16552 Tiggee	611	15169 Google	1,012	39572 Advancedh.	560
4134 ChinaNet	468	4134 ChinaNet	814	39572 Advancedh.	522	26496 GoDaddy	535	24940 Hetzner	873	60781 Leaseweb	478
19551 Incapsula Inc	397	15169 Google	814	12876 AS12876	452	60781 Leaseweb	398	15395 London Off.	822	16552 Tiggee	475
54113 Fastly	361	4837 China169	728	63949 Linode	438	33517 DynDNS	364	36351 Softlayer	753	49505 Selectel	433
63949 Linode	358	54994 Quantil	606	14618 Amazon-2	329	12876 AS12876	354	4808 China Unic.	494	63949 Linode	428
4808 China Unic.	348	35415 Webzilla	551	32475 SingleHop	298	4808 China Unic.	351	20940 Akamai	414	4837 China169	352
(a)		(b)		(c)		(d)		(e)		(f)	

Table 2: Metrics to identify possible attackers and victims: (a) the number of Alexa domains, (b) number of domains hosting JS libraries, (c) number of mailexchange servers, (d) number of name server, (e) number of JS servers whose NS is in a country/AS, and (f) number of MX servers whose NS in a country/AS

For a contry c, we use a similar query and a new rule that propagates the taint from c to all IPs and ASes located in c. The rule is the following:

$$\frac{\mu(n,\mathcal{C}) = \mathbf{c}, e = (m,n) \in E, \lambda(e) = \texttt{LOC\_IN}}{\mu(m,\mathcal{C}) := \mathbf{c}} \qquad (r_{\texttt{LOC}})$$

Metric #2 (Hosted JS Libraries Providers)—The second metric calculates the number of JS hosting servers which are located in an AS or a country. The approach followed is similar to the one illustrated for Metric #1, however, we use a slightly modified version of Rule $r_{\texttt{A}}$:

$$\frac{\Delta, e' = (p,m) \in E, \lambda(e') = \text{JS}}{\mu(m,\mathcal{C}) := \mathbf{c}} \qquad (r_{\texttt{A},\texttt{JS}})$$

where Δ is the precondition of $r_{\texttt{A}}$. The new propositions $e' = (p,m) \in E$ and $\lambda(e') = \text{JS}$ describe the pattern that uniquely distinguishes JS hosting servers from other domains, e.g., a domain hosts a JS program if it has an incoming edge of type `JS`.

Metric #3 (Hosted Email Servers)—The third metric measures the number of email servers hosted by an attacker or victim. The query is similar to Metric #2 in which we modify Rule $r_{\texttt{A}}$ to consider domains with incoming edges of type `MX`.

Metric #4 (Hosted Name Servers)—The fourth metric measures the number of name servers hosted by an attacker or victim. Also, this rule is similar to the previous ones and Rule $r_{\texttt{A}}$ consider domains with incoming edges of type `NS`.

5.2 Second Order Metrics

Starting from the previous metrics, we build more sophisticated ones that quantify the influence of a provider or a country on third-party servers.

Metric #5 (Name Servers for JS Providers)—This metric measures the number of JS hosting servers whose authoritative name servers are hosted by a victim or attacker. The rules used for an AS are $r_{\texttt{ORIG}}$ and the following one:

$$\frac{\cdots, e = (m,n) \in E, \lambda(e) = \texttt{NS}, e' = (p,m) \in E, \lambda(e') = \texttt{JS}}{\mu(m,\mathcal{C}) := \mathbf{c}} \qquad (r_{\texttt{NS},\texttt{JS}})$$

where we used "$\cdots$" as a place holder for the taint precondition. This rule propagates the taint from an AS to its own IPs. An IP is counted if two conditions are met. First the IP n has an incoming edge `NS` from another node m, i.e., n is an authoritative server for m. Second, the node m has an incoming edge of type `JS` from a node p, i.e., m hosts a JS library for p. The query for the case of a country contains the Rule $r_{\texttt{LOC}}$ followed by $r_{\texttt{ORIG}}$ and $r_{\texttt{NS},\texttt{JS}}$.

Metric #6 (Name Servers for Email Servers)—This metric measures the number of domain of email servers whose name server is hosted by a victim/attacker. The construction of the query is the same as for Metric #5. In the case of AS, the rules used are $r_{\texttt{ORIG}}$ and a modified version of $r_{\texttt{NS},\texttt{JS}}$:

$$\frac{\cdots, e = (m,n) \in E, \lambda(e) = \texttt{NS}, e' = (p,m) \in E, \lambda(e') = \texttt{MX}}{\mu(m,\mathcal{C}) := \mathbf{c}} \qquad (r_{\texttt{NS},\texttt{MX}})$$

6. ATTACK EVALUATION

We now evaluate the impact of attacks. We consider three attack scenarios, namely, distribution of JS malicious content (Section 6.1), email sniffing (Section 6.2), and DoS against core service providers (Section 6.3). We present results with two levels of granularity. First, we show the overall impact of attacks in terms of total number of affected Alexa domains. Second, for a selection of attacks, we present attack results on a per-victim base.

6.1 Distribution of JS Malicious Content

For this attack, we consider three techniques: hosting malicious JS content, injection of malicious JS on in-path TCP connections, and malicious name resolution redirection. We select attackers according to metrics #2 and #4 in Table 2. Then, for each technique and attacker, we measure attack results as the number of websites that, as a result of the attack, will distribute the malicious JS content to their users. Tables 3 (a) and 3(b) show the attack results when the attacker is an AS or a country, respectively.

Hosting Malicious JS Content—In this attack we assume that the attacker is either an AS or a country that colluded with web servers hosting JS code. For example, in the case of AS, we assume that the web servers hosted by the AS are cooperating with the origin AS. Possible attackers can be selected with Metric #2, which count the total number of domains hosting JS for each AS or country.

The attack results are shown in Table 3(a). The attack results show that countries can be very powerful attackers. For example, according to Metric #2, the United States hosts 47k JS hosting providers (see Table 2(a)) which could distribute malicious code to about 16% of the top 100k Alexa domains. However, ASes are also very powerful and affect a fraction of websites that is even larger than that of individual countries, and even groups of countries. For example, the AS of Google can affect about 9% of Alexa domains, the number of domains that can be affected by the Netherlands, Russia, Germany, Japan, China and Great Britain combined. Even more interestingly, the AS of Google reaches 9% of websites with only 762 servers compared to 3% of the 10k servers of Amazon. This result highlights that the power of operators can be more precisely measured by taking into account to what extent other services depend on them. The AS of Google is not an isolated case. Other ASes can affect as many domains as a country. Examples of these ASes are CloudFlase, EdgeCast, Amazon-1 and Akamai. Each of them can distribute malicious code to more domains than the top six countries (excluding the United States).

Propagation rules—We created this table with the following rules. When the attacker is the AS, we use Rule $r_{\texttt{ORIG}}$, $r_{\texttt{A}}$, and $r_{\texttt{JS}}$. If the attacker is a country, then we use the Rule $r_{\texttt{LOC}}$ followed by the previous ones, i.e., $r_{\texttt{ORIG}}$, $r_{\texttt{A}}$, and $r_{\texttt{JS}}$. The resulting graph is then processed by the γ function, which counts the number of tainted Alexa domains.

In-path Malicious JS Injection—Interestingly, a very large fraction, i.e., 82% of Table 3(a), of JS hosting service distribute JS libraries over unprotected connections, i.e., HTTP instead of HTTPS. Accordingly, hosting ASes and countries can intercept TCP connections from border gateways and inject malicious content similarly as performed for the Great Cannon attack. We may extend the measurement to protected resources, however, the attacker is required to control a valid certificate for the domain being hijacked. While this is a possible attack scenario, it requires additional effort that, considering the low number of protected resources, will produce a limited increase of the attack result. Table 3(a) shows the attack results on Alexa Web sites that include an unprotected JS program.

Among the 82% of JS inclusion over unprotected connections, 1,079 of them are crossing the Chinese network borders. However, China is not the country that can affect the largest fraction of websites. Other countries could perform better than China including the United States with 12,267 websites, the Netherlands with 2,639 websites, and Russia with 1,409 websites. An interesting aspect of our results is that this type of attack method does not perform any better than the hosting malicious content attack. In fact, injecting malicious JS code via web server collusion affects 17% fewer affected domains on average than hosting malicious content.

Similarly to the attack based on hosting malicious content, we observed that ASes can affect more domains than countries. For example, the AS of Google can affect as many domains as the Netherlands, Russia, and Germany together. However, also in this case, in-path malicious JS injection does not reach as many domains as the injection via server collusion. For example, an in-path code injection can cause Google to lose about 41% of total websites.

Now, we present a fine-grained analysis of this attack. We map the attack results to countries that would be affected if another country decides to perform this attack. An excerpt of these results are presented in Table 4(a). Attack results can be interpreted as a form of dependency among countries. Our results show two interesting facts. First, with different intensity, almost all the popular countries (except for six of them) can attack at least one domain of another country. Second, the dependency among countries is not symmetric. For example, consider the United States. According to all metrics, the United States is the most powerful attacker in our model. However, this influence is not symmetric, e.g., when compared to the Netherlands. While the United States can affect 283 Dutch domains, 967 US domains can be attacked by the Netherlands.

Propagation rules—The rules for this measurement are similar to those of the previous attack. However, we modified Rule $r_{\texttt{JS}}$ to limit the propagation to unprotected JS edges only:

$$\frac{\mu(n,\mathcal{C}) = \mathbf{c}, e = (m,n) \in E, \lambda(e) = \texttt{JS}, \mu(e,\mathcal{S}) = \texttt{HTTP}}{\mu(m,\mathcal{C}) := \mathbf{pc}} \tag{$r_{\texttt{JS}}$}$$

where $\mathcal{S}$ is the property key that stores the URL scheme of the include JS.

Malicious Name Resolution Redirection—Finally, malicious content can be distributed to Web browsers via malicious domain name resolution. In this attack, we assume that the authoritative name server of a domain hosting JS redirects users to a malicious server. The attack result is the number of websites that include a resource hosted on a server whose name server is colluded or compromised. This attack exploits three types of relationships of our model. The first relationship is between domains and the domains hosting JavaScript. The second relationship is the domain name resolution which maps, eventually, domain names to IPs. The third type of relationship is the domain name resolution process with an operator, e.g., country or network provider.

With this technique, countries do not gain considerably more power than the previous attacks. In most of the cases, all players can affect a slightly lower number of websites. Only two players stand out from the rest, i.e., the AS of

Country/AS	Host. coll.	In-path Inj.
United States	15,658	12,267
Netherlands	3,292	2,639
Russia	1,701	1,409
Germany	1,622	1,317
Japan	1,311	1,151
China	1,141	1,079
Great Britain	1,094	895
Ireland	1,048	828
EU	905	824
France	713	603
Canada	399	246
Poland	176	151
Italy	105	97
Spain	83	69
15169 Google	9,469	5,553
13335 CloudFlare	4,310	3,165
15133 EdgeCast	3,404	2,306
16509 Amazon-1	3,216	2,264
20940 Akamai	2,279	1,800
14618 Amazon-2	572	351
35415 Webzilla	515	479
24940 Hetzner	379	330
16276 Ovh	342	287
36351 Softlayer	334	286
4837 China169	227	226
4134 ChinaNet	198	185
37963 Alibaba	148	146
54994 Quantil	71	71

(a)

Country/AS	DNS redir.
United States	12,375
Russia	1,362
Netherlands	1,225
China	1,032
Japan	880
EU	743
Germany	621
France	454
Singapore	317
Great Britain	225
Spain	173
Iran	124
Canada	117
Poland	65
15169 Google	7,859
33517 DynDNS	4,311
16509 Amazon-1	3,685
13335 CloudFlare	3,012
4812 China Tlc	595
4837 China169	555
4808 China Unic	401
16552 Tiggee	361
26496 GoDaddy	316
24940 Hetzner	227
16276 Ovh	199
36351 Softlayer	196
20940 Akamai	88
15395 London Off	88

(b)

Country/AS	MX coll.
United States	24,459
Germany	2,301
Great Britain	1,838
Russia	1,602
France	1,382
Japan	1,317
Netherlands	1,279
Ireland	809
Canada	614
India	496
Spain	410
Iran	392
Italy	384
Turkey	319
15169 Google	11,127
8075 Microsoft	2,465
26496 GoDaddy	1,267
16276 Ovh	565
24940 Hetzner	347
16509 Amazon-1	332
36351 Softlayer	237
60781 Leaseweb	170
12876 AS12876	134
46606 Unified L	113
63949 Linode	108
14618 Amazon-2	104
39572 Advancedh	96
32475 SingleHop	93

(c)

Country/AS	MX+NS
United States	13,077
Netherlands	3,933
Ireland	3,006
China	2,300
Germany	1,405
Great Britain	1,735
Russia	1,466
France	910
Japan	902
Iran	344
Spain	338
Canada	265
Italy	242
Turkey	213
8075 Microsoft	3,003
13335 CloudFlare	2,280
4837 China169	1,784
26496 GoDaddy	1,447
16509 Amazon-1	1,256
33517 DynDNS	1,178
16276 Ovh	555
24940 Hetzner	335
16552 Tiggee	227
36351 Softlayer	179
39572 Advancedh.	96
60781 Leaseweb	75
49505 Selectel	65
63949 Linode	57

(d)

Table 3: Attack evaluation: (a) hosting malicious JS content and in-path malicious JS injection, (b) malicious name resolution (c) email sniffing via malicious email provider, and (d) malicious name resolution for email sniffing

	CN	DE	FR	GB	JP	NL	RU	US	Total
US	134	355	223	172	657	290	287	4,248	6,366
RU	0	107	22	14	0	70	364	124	701
NL	12	53	42	42	189	171	55	979	1,543
JP	7	4	1	1	597	7	0	185	802
GB	79	20	15	49	21	29	9	322	544
FR	0	18	84	10	24	10	17	150	313
DE	77	191	31	14	33	38	62	312	758
CN	475	6	3	0	19	0	1	109	613
Total	784	754	421	302	1,540	615	795	6,429	11,640

(a)

	CN	DE	FR	GB	JP	NL	RU	US	Total
US	1,472	1,060	626	765	455	2,047	229	17,245	23,899
RU	0	179	53	24	0	72	933	268	1,529
NL	5	85	43	72	0	512	21	548	1,286
JP	9	13	0	3	1,149	59	0	112	1,345
GB	9	165	115	766	59	292	16	496	1,918
FR	1	40	918	50	1	66	5	356	1,437
DE	6	1,503	88	144	4	209	37	542	2,533
CN	2,387	31	3	6	18	13	0	281	2,739
Total	3,889	3,076	1,846	1,830	1,686	3,270	1,241	19,848	36,686

(b)

Table 4: Attack results group by countries: (a) JS injection on in-path TCP connections (b) Malicious email providers

Google and DynDNS. Google hosts 73 name servers that can be used to distribute malicious JS to the users of 7,859 domains. This amounts to an increase of 677% of the number of controlled name servers. Similarly, DynDNS controls 3,570 name servers that can affect 4,311 domains. We discuss in detail the role played by DynDNS in our model in Section 6.3.

Propagation rules—We create this table with the following rules. When the attacker is the AS, we use Rule $r_{\mathtt{ORIG}}$, $r_{\mathtt{A}}$, and $r_{\mathtt{JS}}$. If the attacker is a country, then we use the Rule $r_{\mathtt{LOC}}$ followed by the previous ones, i.e., $r_{\mathtt{ORIG}}$, $r_{\mathtt{A}}$, and $r_{\mathtt{JS}}$. The resulting graph is then processed by a γ function that counts the number of tainted Alexa domains. The number of affected domains is showed in Table 3(b). The used propagation rules are $r_{\mathtt{ORIG}}$, $r_{\mathtt{A}}$, $r_{\mathtt{Ns}}$ and $r_{\mathtt{JS}}$, where $r_{\mathtt{Ns}}$ is defined as follows:

$$\frac{\mu(n, \mathcal{C}) = \mathbf{c}, e = (m, n) \in E, \lambda(e) = \mathtt{NS}}{\mu(m, \mathcal{C}) := \mathbf{c}} \qquad (r_{\mathtt{Ns}})$$

6.2 Email Sniffing

To acquire a large number of emails, an attacker can rely on various techniques. In this paper we consider two. The first one is by acquiring them directly from the email server. The second one is by redirecting an email client toward a malicious mail server, which will accept the email, keep a copy, and forward it to the intended recipient. This attack can be performed by a provider or by a country. Tables 3(c) and 3(d) show the attack results. All values are the number of Alexa domains that will be affected by this attack grouped by technique and attacker.

Malicious Email Provider—Attackers that can perform this type of attack are selected using Metric #3, i.e., ASes or countries hosting email servers. This attack technique shows the predominance of the United States and Google in managing the email infrastructure of a large fraction of popular websites (See Table 3(c)). The United States alone can acquire emails of 25% of the most popular websites. Similarly, the AS of Google is hosting only 568 email servers which are used by 11% of the websites. The other players, such as Germany, have still relevant influence but up to 10 times less than the US or Google. Interestingly, most of the domains that can be affected by a country are hosted in the same country. For example 17K of the domains affected by the US are hosted in the US (See Table 4(b)).

Malicious Name Resolution Redirection—Attackers for this attack are selected using Metric #6. This metric measures the number of mail server whose authoritative name server is hosted by the attacker. Starting from this list of attackers, we counted the number of Alexa domains that use one of these mail servers. Table 3(d) shows the total number of affected Alexa domains. As we cannot measure to what extent TLS is used as part of the client-to-MTA or MTA-to-MTA email transfer, we neglect the fact that malicious name servers potentially cannot redirect communication. The numbers provided in this scenario thus rather constitute upper bounds.

With this type of attack, we observe that Google loses most of its power. This can be explained by the fact that websites use Google email server via name servers which are not hosted by Google.

6.3 DoS against Core Service Provider

In this section, we consider the case in which a service provider is the victim of an attack. Here, we do not focus on the specific attack technique, but on the impact of making a provider unavailable. The metrics of Table 2 can be used to select a candidate. The queries used for the attack results can be reused for this type of assessment.

For example, let us consider the DoS attack that on the 21st of October 2016 was launched against Dyn.com. DynDNS is an autonomous system operated by Dyn.com. According to our model based on the top 100K Alexa domains, DynDNS does not host a relevant number of mail servers and JS hosting providers. However, it hosts 364 domain name servers. These name servers are authoritative for 3,570 domains hosting JS that provide JS to 5,559 top 100K Alexa domains (not shown in Table 3), of which 4,331 are unprotected JS inclusion. Furthermore, the name servers hosted by DynDNS are authoritative for 1,523 domains running mail servers which are used by 1,178 top Alexa domains. If the Dyn.com DNS infrastructure is attacked, then a fraction that ranges from 1 to 5% of the top 100K Alexa domains would be affected. The operation of these domains may be severely compromised, as JS used to deliver services via Web applications would no longer be available.

7. LIMITATIONS

We evaluate the impact of the attacks over static data acquired in a single time point, which can be seen as a snapshot of the current network status. Therefore the dependency graph is static. Also, we did not consider different network views (e.g., from different locations) and the data has been collected from a single vantage point located at Saarland University in Germany. This vantage points may have an influence especially for geographically distributed Content Delivery Networks (CDNs), in that our analyses are potentially biased based on the client's residence. An interesting direction for future work would be to include different vantage points. Besides geographic differences based on the Web topologie, we believe that building a dependency graph from different vantage points especially from countries deploying censorship filters could also reveal other intriguing results and change the impact of the attacks.

To acquire JavaScript dependencies between domains, we used Scrapy[4], a framework to extract static data from websites. Scrapy does not support the execution of JavaScript. As a result, our model does not include dependencies that may originate from the execution of the JavaScript program. The extraction of these dependencies can be achieved by using advanced Web crawlers (e.g., jÄk [23]). While this, as shown by prior research, can increase coverage up to 80% [23], it is not a viable option for large-scale analyses. Crawlers that interpret JavaScript are considerably slower than classic crawlers as they require more resources and longer execution time for each website.

8. RELATED WORK

The security of the Internet infrastructure has been under constant scrutiny of the research community. Countless works have been presented by using formal and empirical analyses. For example, Albert et al. [5] studied Internet robustness against random errors and targeted attacks. They show that the Internet provides high error tolerance, but it does not provide adequate robustness against attacks targeting hubs (i.e., nodes with higher connection degree). Following this seminal work, other works assessed other aspects of the problem such as hub selection (e.g., [26, 8]). Following this research line, our work takes an empirical approach to mine topology of the Internet infrastructures to study the impact of large-scale attacks.

The security of the Internet has been studied also empirically with measurements of BGP infrastructure [10], JS inclusion [20], Web service networks [13], and HTTPS ecosystem [7]. Frey et al. [10] presented an analysis on the European BGP backbone using publicly available BGP data. Nikiforakis et al. [20] showed that the vast majority of websites rely on external JS libraries stored on poorly maintained web servers. Finally, Cangialosi et al. [7] studied dependences among providers based on shared X509 certificates, and its implications on the HTTPS ecosystem security. Our work presents a similar *what-if* analysis which complements these papers. However, while these works considered individual service in isolation, our work is more comprehensive, considering different services, intra-services relationships, and a framework to support analyses similar to the aforementioned works.

Finally, another line of works attempts to learn service dependencies via observations of network traffic (e.g., NSDMiner [18] and Rippler [25]). While these tools can effectively learn dependnecies, they require network traffic which is not available for the global analyses like the one presented by our paper and other works (e.g., [7]).

9. CONCLUSION

In this paper, we proposed an investigation techniques to assess global-scale threats. We presented a model of the Internet infrastructures based on property graphs. Moreover, to mine the data from the model, we presented a taint-style propagation technique for traversing the graph. We evaluated our framework, on a model built upon the top 100k Alexa domains by passively and actively collecting publicly available information. Using the presented metrics for selecting attacker and victim candidates, we assessed the impact of the attackers and identified the most influential Internet players. Finally, we showed how one country can influence another by using JS injection on in-path TCP connections and MX server collusion.

Acknowledgment

We thank the anonymous reviewers for their constructive
comments. This work was supported by the German
Federal Ministry of Education and Research (BMBF)
through funding for the Center for IT-Security, Privacy
and Accountability (CISPA) (FKZ: 16KIS0345,
16KIS0656) and the project BOB (FKZ: 13N13250).

[4] See https://scrapy.org/

References

[1] MaxMind: IP Geolocation and Online Fraud Prevention. http://dev.maxmind.com/.

[2] RIPE Stat: Information about specific IP addresses and prefixes. https://stat.ripe.net/.

[3] The New Threat: Targeted Internet Traffic Misdirection. http://research.dyn.com/2013/11/mitm-internet-hijacking/.

[4] UK traffic diverted through Ukraine. http://research.dyn.com/2015/03/uk-traffic-diverted-ukraine/.

[5] R. Albert, H. Jeong, and A.-L. Barabási. Error and attack tolerance of complex networks. *nature*, 406(6794):378–382, 2000.

[6] K. R. Butler, T. R. Farley, P. McDaniel, and J. Rexford. A survey of bgp security issues and solutions. *Proceedings of the IEEE*, 98(1):100–122, 2010.

[7] F. Cangialosi, T. Chung, D. R. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson. Measurement and analysis of private key sharing in the HTTPS ecosystem. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 628–640, 2016.

[8] R. Cohen, K. Erez, D. B. Avraham, and S. Havlin. Breakdown of the Internet under Intentional Attack. *Physical Review Letters*, 86(16):3682–3685, Apr. 2001.

[9] L. Daigle. WHOIS Protocol Specification. RFC 3912 (Draft Standard), Sept. 2004.

[10] S. Frey, Y. Elkhatib, A. Rashid, K. Follis, J. Vidler, N. Race, and C. Edwards. It bends but would it break? topological analysis of bgp infrastructures in europe. In *2016 IEEE European Symposium on Security and Privacy (Euro S&P 16)*, pages 423–438, March 2016.

[11] E. Hjelmvik. China's man-on-the-side attack on github. http://bit.ly/2kx4zAE, 2015.

[12] W. Jiang, D. Lee, and S. Hu. Large-scale longitudinal analysis of soap-based and restful web services. In *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pages 218–225, June 2012.

[13] S.-C. Kil, Hyunyoungand Oh, E. Elmacioglu, W. Nam, and D. Lee. Graph theoretic topological analysis of web service networks. *World Wide Web*, 12(3):321–343, 2009.

[14] S. Landau. Making sense from snowden: What's significant in the nsa surveillance revelations. *IEEE Security Privacy*, 11(4):54–63, July 2013.

[15] S. Liu, I. Foster, S. Savage, G. M. Voelker, and L. K. Saul. Who is .com?: Learning to parse whois records. In *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, IMC '15, pages 369–380, New York, NY, USA, 2015. ACM.

[16] B. Marczak, N. Weaver, J. Dalek, R. Ensafi, D. Fifield, S. McKune, A. Rey, J. Scott-Railton, R. Deibert, and V. Paxson. An analysis of china's "great cannon". In *5th USENIX Workshop on Free and Open Communications on the Internet (FOCI 15)*, Washington, D.C., Aug. 2015. USENIX Association.

[17] G. Nakibly, J. Schcolnik, and Y. Rubin. Website-targeted false content injection by network operators. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 227–244, Austin, TX, Aug. 2016. USENIX Association.

[18] A. Natarajan, P. Ning, Y. Liu, S. Jajodia, and S. E. Hutchinson. NSDMiner: Automated discovery of Network Service Dependencies. In *Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, March 25-30, 2012*, pages 2507–2515, 2012.

[19] J. Newland. Large scale ddos attack on github.com. https://github.com/blog/1981-large-scale-ddos-attack-on-github-com, 2015.

[20] N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. Van Acker, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. You are what you include: Large-scale evaluation of remote javascript inclusions. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 736–747, New York, NY, USA, 2012. ACM.

[21] A. Noroozian, M. Korczyński, C. H. Gañan, D. Makita, K. Yoshioka, and M. van Eeten. *Who Gets the Boot? Analyzing Victimization by DDoS-as-a-Service*, pages 368–389. Springer International Publishing, Cham, 2016.

[22] G. Pellegrino, C. Rossow, F. J. Ryba, T. C. Schmidt, and M. Wählisch. Cashing out the great cannon? on browser-based ddos attacks and economics. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*, Washington, D.C., Aug. 2015. USENIX Association.

[23] G. Pellegrino, C. Tschürtz, E. Bodden, and C. Rossow. *jÄk: Using Dynamic Analysis to Crawl and Test Modern Web Applications*, pages 295–316. Springer International Publishing, Cham, 2015.

[24] D. A. Wheeler and G. N. Larsen. Techniques for cyber attack attribution. Technical report, DTIC Document, 2003.

[25] A. Zand, G. Vigna, R. A. Kemmerer, and C. Kruegel. Rippler: Delay injection for service dependency detection. In *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*, pages 2157–2165, 2014.

[26] J. Zhao, J. Wu, M. Chen, Z. Fang, X. Zhang, and K. Xu. K-core-based attack to the internet: Is it more malicious than degree-based attack? *World Wide Web*, 18(3):749–766, 2015.