The Chain of Implicit Trust: An Analysis of the Web Third-party Resources Loading

Muhammad Ikram
muhammad.ikram@mq.edu.au
Macquarie University
University of Michigan

Rahat Masood
rahat.masood@data61.csiro.au
UNSW and Data61, CSIRO

Gareth Tyson
g.tyson@qmul.ac.uk
Queen Mary University of London

Mohamed Ali Kaafar
dali.kaafar@mq.edu.au
Macquarie University and Data61, CSIRO

Noha Loizon
noha.loizon@data61.csiro.au
Data61, CSIRO

Roya Ensafi
ensafi@umich.edu
University of Michigan

ABSTRACT

The Web is a tangled mass of interconnected services, where websites import a range of external resources from various third-party domains. The latter can also load resources hosted on other domains. For each website, this creates a dependency chain underpinned by a form of implicit trust between the first-party and transitively connected third-parties. The chain can only be loosely controlled as first-party websites often have little, if any, visibility on where these resources are loaded from. This paper performs a large-scale study of dependency chains in the Web, to find that around 50% of first-party websites render content that they did not directly load. Although the majority (84.91%) of websites have short dependency chains (below 3 levels), we find websites with dependency chains exceeding 30. Using VirusTotal, we show that 1.2% of these third-parties are classified as suspicious — although seemingly small, this limited set of suspicious third-parties have remarkable reach into the wider ecosystem.

ACM Reference Format:
Muhammad Ikram, Rahat Masood, Gareth Tyson, Mohamed Ali Kaafar, Noha Loizon, and Roya Ensafi. 2019. The Chain of Implicit Trust: An Analysis of the Web Third-party Resources Loading. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313521

1 INTRODUCTION

In the modern web ecosystem, websites often load resources from a range of third-party domains such as ad providers, tracking services, content distribution networks (CDNs) and analytics services. This is a well known design decision that establishes an *explicit trust* between websites and the domains providing such services. However, often overlooked is the fact that these third-parties can further load resources from other domains, creating a *dependency chain.* This results in a form of *implicit trust* between first-party websites and any domains loaded further down the chain.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313521

Consider the `bbc.com` webpage, which loads JavaScript from `widgets.com`, which, upon execution loads additional content from another third-party, say `ads.com`. Here, `bbc.com` as the first-party website, *explicitly* trusts `widgets.com`, but *implicitly* trusts `ads.com`. This can be represented as a simple dependency chain in which `widgets.com` is at level 1 and `ads.com` is at level 2. Past work tends to ignore this, instead, collapsing these levels into a single set of third-parties [4, 22]. Here, we argue that this overlooks a vital aspect of website design. For example, it raises a notable security challenge, as first-party websites lack visibility on the resources loaded further down their domain's dependency chain. This potential threat should not be underestimated as errant active content (*e.g.,* JavaScript) opens the way to a range of further exploits, *e.g.,* Layer-7 DDoS attacks [23] or massive ransomware campaigns [15].

This paper studies dependency chains in the web ecosystem. Although there has been extensive work looking at the presence of third-parties in general [4, 20, 22], little work has focused on how content is indirectly loaded. We start by inspecting how extensive dependency chains are across the Alexa's top-200K (Section 2). We confirm their prominence, finding that around 50% of websites *do* allow third-parties to form dependency chains (*i.e.,* they implicitly trust third-parties they do not directly load). The most commonly *implicitly* trusted third-parties are well known operators, *e.g.,* `google-analytics.com` and `doubleclick.net`: these are implicitly imported by 68.3% (134,510) and 46.4% (91,380) websites respectively. However, we also observe a wide range of more obtuse third-parties such as `pippio.com` and `51.la` imported by 0.52% (1,146) and 0.51% (1,009) of websites. Although the majority (84.91%) of websites have short chains (with levels of dependencies below 3), we find first-party websites with dependency chains exceeding 30 in length. This not only complicates page rendering, but also creates notable attack surface.

We then proceed to inspect if *suspicious* or even potentially *malicious* third-parties are loaded via these long dependency chains (Section 4). We do not limit this to just traditional malware, but also include third-parties that are known to mishandle user data and risk privacy leaks. Using the VirusTotal API [14], we classify third-party domains into innocuous *vs.* suspicious. When using a reasonable classification threshold, we find that 1.2% of third-parties are classified as suspicious. Although seemingly small, we find that this limited set of suspicious third-parties have remarkable reach. 73% of websites under-study load resources from suspicious third-parties, and 24.8% of first-party webpages contain at least 3 third-parties classified as suspicious in their dependency chain. This, of course, is impacted by many considerations which we explore — most notably, the power-law distribution of third-party popularity, which sees a few major players on a large fraction of websites. We share our datasets, experimental testbed code and scripts used in this paper with the wider research community for further analysis of the consequences of implicit trust `https://wot19submission.github.io`.

2 DATASET AND DATA ENRICHMENT

2.1 Data Collection

We obtain the resource dependencies of the Alexa top-200K websites' main pages[1] using the method described in [18]. This Chromium-based Headless [6] crawler renders a given website and tracks resource dependencies by recording network requests sent to third-party domains. The requests are then used to reconstruct the dependency chains between each first-party website and its third-party URLs. Note that each first-party can trigger the creation of multiple dependency chains (to form a tree structure). To construct the dependency tree, we identify third-party requests by comparing the second level domain of the page (*e.g.,* `bbc.com`) to the domains of the requests (*e.g.,* `cdn.com` and `ads.com` via `widgets.com`). Those with different second level domains are considered third-party. We ignore the sub-domains so that a request to a domain such as `player.bbc.com` is not considered as third-party. Due to the lack of purely automated mechanism to disambiguate between site-specific sub-domains (*e.g.,* `player.bbc.com`) or country-specific sub-domains (*e.g.,* `bbc.co.uk`), we leverage Mozilla Public Suffix list [27] and `tldextract` [19] for this task. From the Alexa Top-200k websites, we collect 11,287,230 URLs which consist of 6,806,494 unique external resources that correspond to 68,828 and 196,940, respectively, unique second level domains of third- and first-parties.

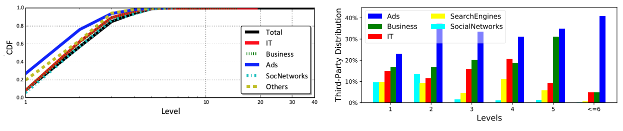
Constructing the dependencies between objects in a webpage is a non-trivial task. In cases where third-party JavaScript gets loaded into a first-party context, and then makes an AJAX request, the HTTP(S) request appears to be from the first-party (i.e. the *referrer* will be the first-party). To overcome such cases and to preserve the information on relations between the nested resource dependencies, we allow the crawler to include the URL of the third-party from which the JavaScript was loaded by first-party.

2.2 Data Enrichment

To augment the dependency data, we use VirusTotal which is an online solution which aggregates the scanning capabilities provided by more than 68 AV tools, scanning engines and datasets. It has been commonly used in the academic literature to detect malicious apps, executables, software and domains [7, 11, 13, 16, 17]. We use the VirusTotal report APIs to obtain the VTscore for each third-party URL. This score represents the number of AV tools that flagged the website as malicious (max. 68). The reports also contain meta-information such as the first scan date, scan history, domain name resolution (DNS) history, website or domain category, reverse DNS, and whois information. We further supplement each domain with their WebSense [31] category provided by the VirusTotal's record

	Alexa Rank					
	1-200K	1-10K	190-200K	10-50K	50-100K	100-200K
F.-Parties that trust:						
All Resources:						
Explicit (Lvl. 1)	95%	95%	95%	94%	95%	95%
Implicit (Lvl. $\geq$ 2)	49.7%	55.1%	47.9%	51.8%	50.23%	48%
JavaScript:						
Explicit	91%	92%	91%	91%	91%	90%
Implicit	49.5%	55%	47.8%	51.69%	50%	47.8%

Table 1: Overview of the Dataset for different ranges of Alexa's ranking. The rows indicate the proportion of Alexa's Top-X websites that explicitly and implicitly trust at least one third-party (*i*) resource (of any type); and (*ii*) JavaScript.



(a) **(b)**

Figure 1: (a) CDF of dependency chain lengths (broken down into categories of first-party websites); and (b) distribution of third-party websites across various categories and levels.

API. During the augmentation, we eliminate repeating, unresponsive or invalid URLs in each dependency chain. Thus, we collect the above metadata for each second level domain in our dataset. This results in a final sample of 196,940 first-party websites, and 68,828 third-party domains.

3 EXPLORING THE CHAINS

We begin by exploring the presence and usage of implicit trust chains. We first confirm if websites do, indeed, rely on implicit trust and then explore how these chains are used.

3.1 Do websites rely on implicit trust?

Overall, the Top-200k dataset makes 11,287,230 calls to 6,806,494 unique external resources, with a median of 27 external resources per first-party website. To dissect this, Table 1 presents the percentage of webpages in each Alexa range that load explicitly and implicitly trusted third-parties. Confirming prior studies [4, 20], it shows that 95% of websites import external resources, with 91% importing externally hosted JavaScript. More important is that observation that around 50% of the websites *do* rely on implicit trust chains, *i.e.,* they allow third-parties to load further third-parties on their behalf. The propensity to form dependency chains is marginally higher in more popular websites; for example, 55% in the Alexa top 10K have dependency chains compared to 48% in the bottom 10K (*i.e.,* rank 190-200K). In other words, more popular websites tend to rely more on implicitly trusted third-parties.

These implicitly trusted third-parties appear at various positions in the dependency chain. Intuitively, long chains are undesirable as they typically have a deleterious impact on page load times [30] and increase attacks surface. Figure 1a presents the CDF of chain length for all first-party websites. For context, websites are separated into their sub-categories.[2] It shows that 80% of the first-party websites create chains of trust of length 3 or below. However, there is also a small minority that dramatically exceed this chain length:

[1]We select the top 200K as this gives us broad coverage of globally popular websites, whilst also remaining tractable for our subsequent data enrichment activities.

[2]We only include the most popular categories.

Lev.	Total	Image	JS	Data	Font/CSS	Uncat.
1	9,212,245	34.4%	30.6%	16.0%	7.8%	11.3%
2	1,566,841	48.8%	16.7%	11.7%	3.3%	19.4%
3	405,390	45.0%	12.3%	11.1%	1.3%	30.2%
4	78,107	41.8%	18.4%	8.0%	8.1%	23.6%
5	14,413	40.6%	18.0%	12.8%	2.0%	26.4%
≥6	10,208	36.6%	12.3%	13.0%	1.2%	36.8%

Table 2: Breakdown of resource types requested by the Top-200K websites across each level in the dependency chain.

we find that all website categories import ≈2% of their external resources from level 3 and above. In the most extreme case, we see `rg.ru` (news) with a chain containing 38 levels, consisting of mutual calls between `adriver.ru` (ad provider) and `admelon.ru` (IT website). Other notable examples include `thecrimson.com.bg` (Harvard's student newspaper), `argumenti.ru` (news), `mundomax.com` (IT news), `lifestyle.bg` (entertainment), which have a maximum dependency level of 15. We argue that these complex configurations make it extremely difficult to reliably audit such websites, as a first-party cannot be assured of which objects are later loaded.

3.2 What objects exist in the chain?

The previous section has confirmed that a notable fraction of websites create dependency chains with (up to) tens of levels. We next inspect the types of resources imported within these dependency chains. We classify resources into four main types: Image, JavaScript, Data (consisting of HTML, JSON, XML, plain text files), and CSS/Fonts. Table 2 presents the volume of each resource type imported at each level in the trust chain. We observe that the make-up of resources varies dramatically based on the level in the dependency chain. For example, the fraction of images imported tend to increase — this is largely because third-parties are in-turn loading images (*e.g.,* for adverts). In contrast, the fraction of JavaScript decreases as the level in the dependency chain increases: 30.6% of resources at level 1 are JavaScript compared to just 12.3% at level 3. This trend is caused by the fact that new levels are typically created by JavaScript execution (thus, by definition, the fraction of JavaScript must deplete along the chain). However, it remains at a level that should be of concern to web engineers as this confirms a significant fraction of JavaScript code is loaded from potentially unknown implicitly trusted domains.

To build on this, we also inspect the *categories* of third-party domains hosting these resources. Figure 1b presents the make-up of third-party categories at each level in the chain. It is clear that, across all levels, advertisement domains make up the bulk of third-parties. We also notice other highly demanded third-party categories such as search engines, Business and IT. These are led by well known providers, *e.g.,* `google-analytics.com` (web-analytics[3]) is on 68.3% of pages. The figure also reveals that the distributions of categories vary across each dependency level. For example, 23.1% of all loaded resources at level 1 come from advertisement domains, 37.3% at level 2, 46.2% at level 3, *i.e.,* the proportion increases across dependency levels. In contrast, social network third-parties (*e.g.,* Facebook) are mostly presented at level 1 (9.58%) and 2 (13.57%) with a significant drop at level 3. The dominance of advertisements is not, however, caused by a plethora of ad domains: there are far fewer ad domains than business or IT (see Table 3). Instead, it is driven by the large number of requests to advertisements: Even though ad domains only make-up 1.5% of third-parties, they generate 25% of resource requests. Importantly, these popular providers can trigger further dependencies; for example, `doubleclick.com` imports 16% of its resources from further implicitly trusted third-party websites. This makes such domains an ideal propagator of malicious resources for any other domains having implicit trust in it.

4 FINDING SUSPICIOUS CHAINS

We next study the existence of *suspicious* third-parties, which could lead to abuse of the implicit trust. Within this section we use the term *suspicious* (to be more generic than malicious) because VirusTotal covers activities ranging from low-risk (*e.g.,* sharing private data over unencrypted channels) to high-risk (malware).

4.1 Do chains contain suspicious parties?

First, we inspect the fraction of third-party domains that trigger a warning by VirusTotal. From our third-party domains, 2.5% have a VTscore of 1 or above, *i.e.,* at least one virus checker classifies the domain as suspicious. If one treats the VTscore as a ground truth, this confirms that popular websites *do* load content from suspicious third-parties via their chains of trust. However, we are reticent to rely on VTscore $\geq$ 1, as this indicates the remaining 67 virus checkers did not flag the domain.[4] Thus, we start by inspecting the presence of suspicious third-parties using a range of thresholds.

Table 3 shows the fraction of third-parties that are classified as suspicious using several VTscore thresholds. For context, we separate third-parties into their respective categories (using WebSense). The table confirms that a noticeable subset of suspicious third-party domains exist; for example, if we classify any resource with a VTscore $\geq$ 10 as suspicious, we find that 1.2% of third-party domains are classified as suspicious with 6.2% of all resource calls in our dataset going to these third-parties. Notably this only drops marginally (to 5.7%) with a *very* conservative VTscore of $\geq$ 40. We observe similar results when considering thresholds in the [3..50] range. This confirms, with a high certainty, that approximately 6% of resource calls in the dependency chains are towards domains that engage in suspicious activity (further detailed in [12]). We will conservatively refer to domains with a VTscore $\geq$ 10 as suspicious in the rest of this analysis.

We also check if dependency chains contain suspicious JavaScript resources. We focus on JavaScript as active content poses greater threats, *e.g.,* cross-site scripting (XSS) and advanced phishing [20]. Table 4 shows the top first-party domains, ranked according to the number of unique suspicious third-parties in their chain of dependency. We note that the top ranked (most vulnerable) first-party domains belong to various categories such as Content Sharing, News, or IT. This indicates that there is no one category of domains that inherits suspicious JavaScript. However, we note that first party websites categorized as "Business" represent the majority of most exposed domains at Level ≥2, with 16% of the total number of first-party domains implicitly trusting suspicious JavaScript belonging

[3]Grouped as in business category as per VirusTotal reports.

[4]Diversity is likely caused by the virus databases used by the different virus checkers [2]

				VTScore ≥ 3		VTScore ≥ 10		VTScore ≥ 20		VTScore ≥ 40		VTScore ≥ 55	
Category	**Third-Parties**	**Total Calls**	**Suspicious JS**	Num.	Vol.	Num.	Vol.	Num.	Vol.	Num.	Vol.	Num.	Vol.
All	68,828	11,287,204	270,758 (2.4%)	1.6%	6.4%	1.2%	6.2%	1.0%	6.1%	0.6%	5.7%	$\leq$ 0.1%	$\leq$ 0.1%
Business	6,786	1,924,591	184,360 (9.6%)	1.5%	21.5%	1.1%	21.5%	1.0%	21.4%	0.5%	20.6%	0%	0%
Ads	1,017	2,870,482	7,924 (0.3%)	3.5%	0.1%	3.3%	0.1%	2.9%	0.1%	1.6%	$\leq$ 0.1%	0%	0%
IT	8,619	1,646,287	10,547 (0.6%)	2.2%	3.8%	1.5%	3.6%	1.2%	3.5%	0.6%	3.0%	$\leq$ 0.1%	$\leq$ 0.1%
Other	52,406	4,845,844	67,927 (1.4%)	1.4%	4.6%	1.1	4.3%	0.9%	4.2%	0.6%	3.8%	$\leq$ 0.1%	$\leq$ 0.1%

Table 3: Overview of suspicious third-parties in each category. Col.2-4: number of third-party websites in different categories, the number of resource calls to resources, and the proportion of calls to suspicious JavaScript. Col.5-9: Fraction of third-party domains classified as suspicious (*Num.*), and fraction of resource calls classified as suspicious (*Vol.*), across various VTscores (i.e., ≥ 3 and ≥ 55).

to the Business Category, with distant second being the "News & Media" Category and third the "Adult" category. The number of suspicious JavaScript codes loaded by these first-party domains ranges from 4 to 31. We note the extreme case of `amateur-fc2.com` website *implicitly* importing 31 unique suspicious JavaScript programs from 4 unique suspicious domains. Moreover, we observe at most 7 unique third-parties (combining both explicit and implicit level) that is a cause of suspicious JavaScripts in first-parties. This happens for `privet-rostov.ru` domain, having third-party domains such as `charter.com, vk.com. rambler.ru, doubleclick.net, dx.com, cdn.adlegend.com, syncsw.pool.datamind.ru`.

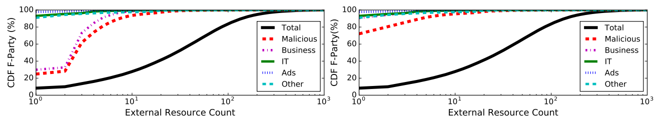
#	First-party Domain	Alexa Rank	# Mal. JSes	Unique Susp. Doms.	Category	Chain Len.
Unique Suspicious Domains at Level = 1						
1	theinscribermag.com	46,242	6	5	Blogs	5
2	skynet-system.com.ua	192,549	6	5	Busin.	4
3	nodwick.com	194,823	13	4	Enter.	4
4	iphones.ru	12,045	4	4	IT	4
5	privet-rostov.ru	193,024	6	4	LifeStyle	4
Unique Suspicious Domains at Level ≥ 2						
1	traffic2bitcoin.com	33,513	6	5	Games	7
2	radionetplus.ru	166,003	8	4	SW Download	6
3	studiofow.tumblr.com	85,483	11	4	Adult	4
4	amateur-fc2.com	52,556	31	4	Adult	5
5	fasttorrent.ru	24,250	9	4	File Sharing	7

Table 4: Top 5 most exposed first-party domains (with VTscore $\geq$ 10) ranked by the number of unique suspicious domains.

4.2 How widespread are suspicious parties?

We next inspect how widespread these suspicious third-parties are at each position in the dependency chain, by inspecting how many websites utilize them. Figure 2a displays the cumulative distribution (CDF) of resource calls to third-parties made by each first-party webpage in our dataset. Within the figure, we decompose the third-party resources into various groups (including total *vs.* suspicious). The figure reveals that suspicious parties within the dependency chains are commonplace: 24.8% of all first-party webpages contain at least 3 third-parties classified as suspicious in their dependency chain. Remarkably, 73% of first-party websites load resources from third-parties at least once. Hence, even though only 1.6% of third-party domains are classified as suspicious, their reach covers nearly three quarters of websites (indirectly via implicit trust).

The above is demonstrated in Table 5, which presents the top 10 most frequently encountered suspicious third-party domains that are providing suspicious JavaScript. It can be seen that popular third-party domains exist across *many* first-party sites. The top 20% of third-party domains cover 86% (9,650,582) of all resource calls. Closer inspection shows that it is driven by one prominent third-party: `google-analytics.com`. At first, we thought that this was an error, however, during the measurement period `google-analytics.com` obtained a VTscore of 51, suggesting a high degree of certainty. This was actually caused by `google-analytics.com` loading another third-party, `sf-helper.net`, which is known to distribute adwares and spywares. It is unclear why Google was performing this. We therefore repeated these checks in October 2018, to confirm that this activity has ceased, and `sf-helper.net` is no longer loaded. To understand the impact its new de-classification has, Figure 2b shows the distribution of resource calls to third-party categories when `google-analytics.com` is benign. This reduces the number of first-party websites exposed to suspicious resources by 63%. This highlights effectively the impact of high centrality third-parties being permitted to load further resources: the infection of just one can immediately effect a significant fraction of websites.

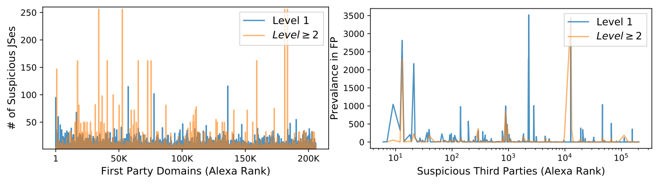


(a) All websites **(b) Without google-analytics**

Figure 2: CDF of resources loaded per-website from various categories of third-parties.

#	Third-party Domain	Alexa Rank	# FP	Category
Prevalence of Third-parties at Level = 1				
1	google-analytics.com	13,200	43,156	Business (Web Analytics)
2	gravater.com	2,292	3,520	IT
3	charter.com	12,714	3,425	Business
4	vk.com	13	2,815	Social Network
5	statcounter.com	2,265	2,327	Business (Web Analytics)
Prevalence of Third-parties at Level ≥ 2				
1	charter.com	12,714	3,452	Business
2	vk.com	13	2,290	Social Network
3	livechatinc.com	888	851	Web Chat
4	onesignal.com	950	467	Business
5	rambler.ru	291	370	SearchEngine

Table 5: Top 5 most prevalent suspicious third-party domains (with VTscore $\geq$ 10) on level 1 (explicit trust) and beyond (implicit trust) providing resources to first-parties. #FP refers to the number of First-party domains having the corresponding suspicious third-party domain in their chain of dependency.



(a) (b)

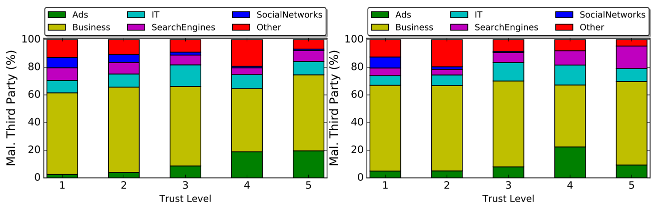
Figure 3: Figure (a) depicts the number of suspicious JavaScript content imported (explicitly and implicitly) by first-party domains shown according to their Alexa ranking; and (b) shows the number of impacted first-party domains as function of the ranking of domains of Suspicious JavaScript.

4.3 How popular are suspicious third-parties?

We next test if widespread suspicious third-parties are also highly ranked within Alexa. We treat this as a proxy for global popularity. Beyond `google-analytics.com` we find several other suspicious third-party domains from the Top 100 Alexa ranking. For-instance, `vk.com`, a social network website mostly geared toward East European countries has been used by 3,094 first-parties and is ranked 13 by Alexa. This website is found to be one of the most prevalent suspicious third-party domains at both level 1 and levels ≥ 2. An obvious reason for this domain's presence is because of other infected (malware-based) apps that try to authenticate users from such domains [24]. Other websites such as `statcounter.com` or `gravater.com` are also among the most prevalent third party domains in level 1. These websites were reported to contain malware in their Javascript codes [3]. For instance, users in statcounter forums reported it as malicious because a Javascript running its website redirects users to a malware website `gocloudly.com`, and forces users to click the button [5].

More generally, we observe the presence of a wide range of Alexa ranks in the list of most prevalent domains at levels ≥ 2. In Figure 3a, we show the number of suspicious JavaScripts imported by the first-party domains (Y-axis) according to their Alexa rank (X-axis). Overall, first-party domains import a larger number of suspicious third-party JavaScript codes at levels ≥ 2, however, the first-party domains seem to be equally vulnerable to the implicit import of suspicious content regardless of their rank. There are exceptions though, signified by the peaks in the number of suspicious JavaScripts — these are near exclusively driven by a large number of ≥level-2 scripts (implicit trust). We also encounter an interesting case, which we exclude from the graphs for readability purposes: The first-party domain `kikar.co.il` imports 2,592 JavaScript codes originating from the third-party `hwcdn.net`, a well-known browser hijacker that has been reported to force users to visit spam pages [29]. The VirusTotal API indicates a VTscore of 22 for this suspicious domain. We also note that 35 other first-party domains have this domain in their chain of dependency. Again, this highlights the risk of implicit trust.

In Figure 3b we show the number of impacted first-party domains as a function of the Alexa Rank of suspicious third-party domains (limited to a maximum Alexa Rank of 1 million) — note the log scale of X-axis. We observe that some very prevalent third-parties have a high Alexa ranking (even excluding `google-analytics.com`). For instance, note a spike around the 2000 rank, which reaches a prevalence of 3500 first-party domains at level 1. This spike is caused by `gravatar.com`, propagating suspicious Javascripts. This supports our statements earlier (from Table 5) where `gravatar.com` is ranked second top most suspicious domain. Similarly, a spike around 10K rank indicates the presence of `charter.com` both at level 1 and 2 respectively. These findings demonstrate the wide variety of third-party suspicious JavaScript content loaded from various, not necessarily "obscure", third-party domains.



(a) All websites **(b) News Websites**

Figure 4: Distribution of calls to suspicious third-party websites per category at each level, for all top-200K websites (Figure 4a) and most vulnerable first-party category (Figures 4b).

	All		**News**		**Sports**		**Entertainment**		**Forums**	
Lv.	All	JS	All	JS	All	JS	All	JS	All	JS
1	61.30%	57.70%	75.40%	73.50%	75.70%	73.20%	69.30%	65.60%	67.40%	65.50%
2	5.20%	2.20%	13.40%	5.60%	11.10%	3.70%	8.60%	4.10%	9.10%	4.05%
3	1.30%	0.18%	2.90%	0.45%	3.60%	0.28%	2.70%	0.30%	3.20%	0.15%
4	0.22%	≤ 0.1%	0.64%	0.08%	0.80%	≤ 0.1%	0.70%	0.08%	0.60%	0.00%
≥ 5	≤ 0.1%	0	0.002	≤ 0.1%	0.001%	≤ 0.1%	0.002%	≤ 0.1%	≤0.001%	0.00%

Table 6: Proportion of top-200K websites importing resources classified as suspicious (with VTscore ≥ 10) at each level.

4.4 At which level do suspicious third-parties occur?

Next, we inspect the location(s) in the dependency chain where these suspicious third-parties are situated. This is vital, as implicitly trusted (≥level 2) resources are far more difficult for a first-party administrator to remove. Table 6 presents the proportion of websites that import at least one resource with a VTscore ≥ 10. We separate resources into their level in the dependency chain. The majority of resources classified as suspicious are located at level 1 in the dependency chain (*i.e.,* they are explicitly trusted by the first-party). 73% of websites containing suspicious third-parties are "infected" via level 1. This suggests that these website operators are not entirely diligent in monitoring their third-party resources. This might include websites that purposefully utilize such third-parties [8]. Perhaps more important, the above leaves a significant minority of suspicious resources imported via *implicit* trust (*i.e.,* level ≥ 2). In these cases, the first-party is potentially unaware of their presence. The most vulnerable category is news: over 15% of news sites import *implicitly* trusted resources from level 2 with a VTscore ≥ 10. Notably, among the 56 news websites importing suspicious JavaScript resources from trust level 3 and deeper, we find 52 loading advertisements from `adadvisor.net`. One possible reason is that ad-networks could be infected or victimized with malware to perform malvertising [21, 28].

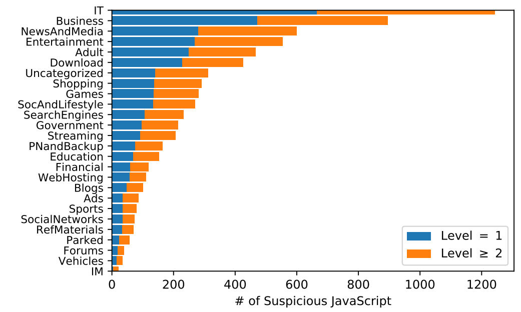


Figure 5: Breakdown of JavaScript resources based on category of domain. Uncategorized category includes domain such as `newmyvideolink.xyz` **and** `cooster.ru`

Similar, albeit less extreme, observations can be made across Sports, Entertainment, and Forum websites. Briefly, Figure 4 displays the categories of (suspicious) third-parties loaded at each level in the dependency chain — it can be seen that the majority are classified as business. This is, again, because of several major providers classified as suspicious such as `convexity.net` and `charter.com`. Furthermore, it can be seen that the fraction of advertisement resources also increases with the number of levels due to the loading of further resources (*e.g.,* images).

Figure 5 also presents the breakdown of the domain categories specifically for suspicious JavaScript resources. Clear trends can be seen, with IT (e.g., `dynaquestpc.com`), Business (*e.g.,*`vindale.com`), News and Media (e.g., `therealnews.com`), and Entertainment (*e.g.,*`youwatchfilm.net`) dominating. Clearly, suspicious JavaScript resources cover a broad spectrum of activities. Interestingly, we observed that 70% and 67%, respectively, of Business (Web analytics) and Ads JavaScripts are loaded from level ≥ 2 in contrast to 17% and 31% of JavaScripts of Government and Shopping loaded at level 1. We next strive to quantify the level of suspicion raised by each of these JavaScripts. Intuitively, those with higher VTscores represent a higher threat as defined by the 68 AV tools used by VirusTotal. Hence, Figure 6 presents the cumulative distribution of the VTscores for all JavaScript resources loaded with VTscore > 0. We separate the JavaScripts into their location in the dependency chain. Clear difference can be observed, with level 2 obtaining the highest VTscore (median 28). In fact, 78% of the suspicious JavaScript resources loaded on trust level 2 have a VTscore > 52 (indicating *very* high confidence). It is also worth noting that the VTscore for resources loaded further down the dependency chain is lower (*e.g.,* level 4). For example, 80% of level 4 resources receive a VTscore below 5. This suggests that the activity of these resources is more contentious, with a smaller number of AV tools reaching consensus. It is impossible to state the reason for this, hence in our extended work [12] we analyze the dynamic activities of these JavaScript content.

5 RELATED WORK

There has been a wealth of research into the utilisation and exploitation of third-parties and JavaScript libraries [9, 10, 20, 22, 25, 26]. Our work differs quite substantially from these in that we are not interested in the JavaScript code itself, nor the simple presence of third-party domains in a webpage. Instead, we are interested in *how* third-parties are loaded, and their use of "implicit" trust (*i.e.,*

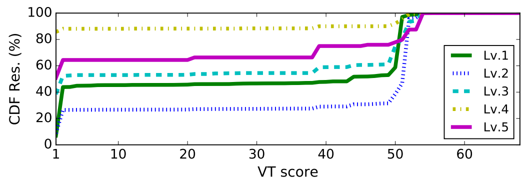


Figure 6: CDF of suspicious JavaScripts (VTscores ≥ 10) at different levels in the chain.

dependency chains). In contrast to our work, these prior studies ignore the presence of dependency chains and treat all third-parties as "equal", regardless of where they are loaded in the dependency chain. Closer to our own work is Bashir *et al.* [1], who studied websites' resource inclusion trees and analyzed retargeted ads using crowdsourcing. This allowed them to identify and classify ad domains, as well as predominant cookie matching partners in the ad exchange environment. Our study is far broader, and sheds light on dependency chains across many different types of websites rather than simply inspecting advertisements. More related is Kumar *et al.* [18], who recently characterized websites' resource dependencies on third-party services. In-line with our work, they found that dependency chains are widespread. This means, for example, that 55% of websites are prevented from fully migrating to HTTPS by their dependencies. Their focus was not, however, on identifying suspicious or malicious activities. To the best of our knowledge, this paper is the first study to analyze the role of implicit trust from a security perspective to better understand the role of dependency chains in loading suspicious third-party content.

6 CONCLUDING REMARKS

This paper has explored dependency chains in the web ecosystem. Inspired by the lack of prior work focusing on how resources are loaded, we found that over 40% of websites *do* rely on implicit trust. Although the majority (84.91%) of websites have short chains, we found first-party websites with chains exceeding 30 levels. Of course, the most commonly *implicitly* trusted third-parties are well known operators (*e.g.,* `doubleclick.net`), but we also observed various less known implicit third-parties. We hypothesized that this might create notable attack surfaces. To confirm this, we classified the third-parties using VirusTotal to find that 1.2% of third-parties are classified as potentially malicious. These resources have remarkable reach — largely driven by the presence of highly central third-parties, *e.g.,* `google-analytics.com`. With this in mind, in our extended work [12], we perform sandbox experiments on the suspicious JavaScript to understand their actions. We witness extensive download activities, much of which consisted of *dropper* files and malware, which is installed on the machine. It was particularly worrying to see that JavaScript resources loaded at level ≥ 2 in the dependency chain tend to have more aggressive properties, particularly as exhibited by their higher VTscore. This is only the first step in our research agenda. We wish to perform longitudinal measurements to understand how these metrics of maliciousness evolve over time. We are particularly interested in understanding the (potentially) ephemeral nature of threats.

REFERENCES

[1] M. A. Bashir, S. Arshad, W. Roebertson, and C. Wilson. Tracing information flows between ad exchanges using retargeted ads. In *USENIX Security Symposium*, 2016.

[2] J. Canto, M. Dacier, E. Kirda, and C. Leita. Large scale malware collection: lessons learned. In *IEEE SRDS Workshop on Sharing Field Data and Experiment Measurements on Resilience of Distributed Computing Systems*. Citeseer, 2008.

[3] I. X. Exchange. Statcounter session hijack. https://exchange.xforce.ibmcloud.com/vulnerabilities/20506, 2005.

[4] M. Falahrastegar, H. Haddadi, S. Uhlig, and R. Mortier. Anatomy of the third-party web tracking ecosystem. *Traffic Measurements Analysis Workshop (TMA)*, 2014.

[5] S. C. Forum. http://www.statcounter.com/ counter/counter.js has malware inside it ! https://forum.statcounter.com/threads/http-www-statcounter-com-counter-counter-js-has-malware-inside-it.43792/, 2016.

[6] Google. Headless chromium. https://chromium.googlesource.com/chromium/src/+/lkgr/headless/README.md, 2018.

[7] D. Ibosiola, I. Castro, G. Stringhini, S. Uhlig, and G. Tyson. Who watches the watchmen: Exploring complaints on the web. In *Web Conference*, 2019.

[8] D. Ibosiola, B. Steer, A. Garcia-Recuero, G. Stringhini, S. Uhlig, and G. Tyson. Movie pirates of the caribbean: Exploring illegal streaming cyberlockers. *International AAAI Conference on Web and Social Media (ICWSM)*, 2018.

[9] M. Ikram, H. Asghar, M. A. Kaafar, and A. Mahanti. On the intrusiveness of javascript on the web. In *CoNEXT Student Workshop*, 2014.

[10] M. Ikram, H. J. Asghar, M. A. Kaafar, A. Mahanti, and B. Krishnamurthy. Towards seamless tracking-free web: Improved detection of trackers via one-class learning. *Proceedings on Privacy Enhancing Technologies*, 2017(1):79–99, 2017.

[11] M. Ikram and M. A. Kaafar. A first look at mobile ad-blocking apps. In *Network Computing and Applications (NCA), 2017 IEEE 16th International Symposium on*, pages 1–8. IEEE, 2017.

[12] M. Ikram, R. Masood, G. Tyson, M. A. Kaafar, N. Loizon, and R. Ensafi. The chain of implicit trust: An analysis of the web third-party resources loading. *arXiv preprint arXiv:1901.07699*, 2019.

[13] M. Ikram, N. Vallina-Rodriguez, S. Seneviratne, M. A. Kaafar, and V. Paxson. An analysis of the privacy and security risks of android vpn permission-enabled apps. In *IMC*, 2016.

[14] V. Inc. Virustotal public api. https://www.virustotal.com/en/documentation/public-api/, 2017.

[15] S. Jerome. Large angler malvertising campaign hits top publishers. https://blog.malwarebytes.com/threat-analysis/2016/03/large-angler-malvertising-campaign-hits-top-publishers/. Accessed: 2018-09-18.

[16] A. Kantchelian, M. C. Tschantz, S. Afroz, B. Miller, V. Shankar, R. Bachwani, A. D. Joseph, and J. D. Tygar. Better malware ground truth: Techniques for weighting anti-virus vendor labels. In *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, pages 45–56. ACM, 2015.

[17] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda. Cutting the gordian knot: A look under the hood of ransomware attacks. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 3–24. Springer, 2015.

[18] D. Kumar, Z. Ma, A. Mirian, J. Mason, J. A. Halderman, and M. Bailey. Security Challenges in an Increasingly Tangled Web. In *Proceedings of the 2017 World Wide Web Conference on World Wide Web*, 2017.

[19] J. Kurkowski. Accurately separate the TLD from the registered domain and subdomains of a url, using the public suffix list. https://github.com/john-kurkowski/tldextract, 2018.

[20] T. Lauinger, A. Chaabane, S. Arshad, W. Robertson, C. Wilson, and E. Kirda. Thou shalt not depend on me: Analysing the use of outdated javascript libraries on the web. In *NDSS*, 2017.

[21] Z. Li, K. Zhang, Y. Xie, F. Yu, and X. Wang. Knowing your enemy: understanding and detecting malicious web advertising. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 674–686. ACM, 2012.

[22] 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 *CCS*, 2012.

[23] G. Pellegrino, C. Rossow, F. J. Ryba, T. C. Schmidt, and M. WÃďhlisch. Cashing out the great cannon? on browser-based ddos attacks and economics. In *USENIX*, 2015.

[24] B. Popa. 85 infected android apps stealing social network passwords found on play store. https://news.softpedia.com/news/85-infected-android-apps-stealing-social-network-passwords-found-on-play- store-518984.shtml, 2017.

[25] J. Su, Z. Li, S. Grumbach, M. Ikram, K. Salamatian, and G. Xie. Web tracking cartography with dns records. In *IEEE 37th International Performance Computing and Communications Conference (IPCC)*, 2018.

[26] J. Su, Z. Li, S. Grumbach, M. Ikram, K. Salamatian, and G. Xie. A cartography of web tracking using dns records. *Computer Communications*, 134:83 – 95, 2019.

[27] M. P. Suffix. View the public suffix list. https://publicsuffix.org/list/, 2018.

[28] A. VANCE. Times web ads show security breach. https://www.nytimes.com/2009/09/15/technology/internet/15adco.html, 2009.

[29] Q. R. Virus. How do i remove hwcdn.net from my pc. https://quickremovevirus.com/how-do-i-remove-hwcdn-net-from-my-pc/, 2017.

[30] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall. Demystify page load performance with wprof. In *Proc. of the USENIX conference on Networked Systems Design and Implementation (NSDI)*, 2013.

[31] Websense. Real-time threat analysis with csi: Ace insight. https://csi.websense.com/, 2018.