REdiREKT: Extracting Malicious Redirections from Exploit Kit Traffic

REdiREKT: Extracting Malicious Redirections from Exploit Kit Traffic

Jonah Burgess*,Domhnall Carlin†,Philip O'Kane‡ and Sakir Sezer§
Centre for Secure Information Technologies, Queen's University, Belfast, Northern Ireland
Email: *jburgess03@qub.ac.uk, †d.carlin@qub.ac.uk, ‡p.okane@qub.ac.uk, §s.sezer@qub.ac.uk

***Abstract*—This paper proposes REdiREKT, a system which utilises the open-source Zeek Intrusion Detection System (IDS) to map HTTP redirection chains observed in Exploit Kit (EK) attacks and extracts distinguishing features to assist machine learning (ML). We build a ground-truth dataset of EK samples, ensuring that the redirection chains for every sample are accurate and reusable in future experiments. By processing a unique combination of 9 redirection techniques, REdiREKT was able to correctly extract 96.52% of malicious domains from 1279 EK samples, spanning 28 families and 8 campaigns, and, only failed to extract 0.7% of malicious chains.**

Using the VirusTotal API to filter out domains flagged as malicious, we build a benign dataset from the Alexa top 10k websites, extracting 12,783 domains from 5910 redirection chains. The malicious redirection data is divided into yearly and family-based categories and compared to the benign results. Based on our analysis of the collected data, we extract and store 48 key features from websites within the redirection chains that could aid future ML-based detection efforts. Finally, we evaluate the performance of REdiREKT, compare it with existing research, and, suggest use-cases and future areas of work.

***Index Terms*—Exploit Kits, Web Security, Malware**

I. Introduction

Web-based malware has plagued the internet for well over a decade. Malicious website observations continue to increase and, in 2019, researchers found that 1 in 10 URLs tested were malicious, and web attacks had increased by 56% since the previous year [1]. Web-based attacks commonly observed in the wild include Exploit Kits (EK), ClickJacking, Phishing and Fake-updaters. CryptoJacking took the spotlight from EKs as the primary threat in recent years [2], but this trend has shifted to FormJacking as a 90% drop in the value of Monero led to a 52% decrease of CryptoJacking throughout 2018 [1].

Although the trends of web-based malware can shift rapidly, attacks often share some common characteristics. They typically utilise a variety of methods to lure victims to malicious websites, use client/server-side scripts to perform malicious functions, obfuscate these scripts to avoid detection, use chains of HTTP redirections to obscure the structure of their malicious ecosystem and implement evasion techniques to slow down defenders. In this paper, we investigate a renowned category of web attacks; Exploit Kits. We conduct our research with a specific focus on the redirection chains used by EKs.

The remainder of this paper is organised as follows. Section II provides a background into Exploit Kits and malicious redirections, while Section III surveys related works. In Section IV, we describe our experimental methodology, present and evaluate our results, and, discuss system limitations. We conclude in Section V and suggest areas of future work.

II. Background

An EK is a malicious software package that can be used to automate the exploitation of computer systems. EKs typically exploit vulnerabilities in web browsers and their related plugins to deliver a malicious payload. These attacks are known as drive-by download (DBD) attacks [3]; they occur silently in the browser when a victim accesses a malicious website.

A. Workflow

The typical EK workflow can be broken down into the following five stages:

1) **Traffic Generation:** To achieve as much traffic as possible, EK operators typically use the following methods:
   - **Compromise Legitimate Websites:** inject malicious code to perform a redirection to an EK.
   - **Malvertising Attacks:** insert malicious code into an advertisement hosted on a popular website.
   - **Spam Email Campaigns:** email the URL to many users hoping some of them will click the link.

   These techniques are often combined with search engine optimisation (SEO) poisoning to boost traffic further.
2) **Redirections:** When a victim visits a compromised site or is served a malicious advertisement, their browser is redirected through a series of intermediate pages known as gates. This chain of redirections obscures the final URL that the victim will arrive at; the landing page.
3) **Fingerprinting:** When a victim arrives at the landing page, the fingerprinting process typically begins (sometimes occurs during the redirection phase too). Client/server-side code is used to identify information about the victim machine, e.g. operating system (OS), browser, plugins, IP address, language and geolocation. The EK uses this data to determine if the system is vulnerable and select the relevant exploit to use.
4) **Exploitation:** If the EK identified a vulnerability for which it has a corresponding exploit, it now attempts to execute it. If multiple vulnerabilities were discovered, it may queue a series of exploits and execute each one until it successfully compromises the system.
5) **Payload Delivery:** If the exploitation phase was successful, the EK executes its payload. Typically, this payload fetches a malicious binary, stores it on the

victim machine and then executes it. The binary could be any variety of traditional malware such as ransomware, keylogger, botnet, trojan, crypto-miner etc.

B. Evasion and Anti-Analysis Techniques

*1) **Obfuscation**:* EKs obfuscate their code to prevent detection by security products. Up to 95% of EKs apply some form of obfuscation to their JavaScript (JS) [4]. Common obfuscation techniques include randomisation, e.g. random whitespace and comments, number obfuscation, e.g. expressing equations in different formats, string obfuscation, e.g. URL/Unicode/hex encoding, string concatenation, character substitution etc. [5].

Some EKs use commercial obfuscation software such as IonCube and ZendGuard [3]. EK authors can test their code against anti-virus (AV) products to determine whether it remains undetectable and update their obfuscation algorithm accordingly. In 2015, researchers at Talos showed that Angler was providing unique payloads for each IP address [6]. Although the functionality was identical, the hashes were different, and, variable and function names were randomised.

*2) **Blacklist Avoidance**:* To avoid being blacklisted, EKs often create new domain names automatically using a domain generation algorithm (DGA). This may be done periodically or in response to the disclosure of URLs/IPs to known blacklists. Recently, EKs started to use 'Domain Shadowing' where they hijack domain registrant accounts and create large numbers of seemingly random subdomains. This technique has proved to be effective against URL blacklisting. In 2015, Cisco researchers identified around 10,000 of such domains [7].

*3) **Cloaking**:* One of the most crucial evasion techniques used by EKs relates directly to the fingerprinting process and makes detection challenging. Many campaigns only target specific countries; if the IP address or system language indicates that the potential victim doesn't meet the criteria, the site won't serve the exploit or reveal malicious code, URLs etc. In this situation, they will typically display a blank page, a 404 not found error or redirect to a benign website [8].

Similarly, if the fingerprinting effort determines that the system is not vulnerable to any of the available exploits, or, the IP address of the system is associated with a security vendor, it will serve a benign page to avoid detection. These evasion techniques require researchers to develop realistic detection systems, often in the form of high-interaction honeypots [4], [9], [10]. The situation is complicated further by the fact that modern EKs only serve an exploit to a victim once. To observe the EKs malicious nature multiple times, a new IP address must be acquired after each attempted infection [8].

C. Business Model

With the rise of EKs, an exploit-as-a-service economy emerged, separating host compromise from monetization [11]. EK authors track infection statistics, allowing them to offer subscriptions on a pay-per-install model where EK operators only pay for successful infections. This subscription-based model encourages innovation; authors protect their source-code, develop user-friendly interfaces, invest in new exploits, improve evasion techniques, and, offer live customer support to compete against rival EK families.

D. Mitigations

AV and anti-malware products help to keep users protected. However, they often rely on signature-based detection, which is only effective against known malware. EKs have been known to integrate the occasional zero-day exploit [12] and modifying code to bypass signature detection is a trivial task which can be automated. Security vendors/researchers maintain blacklists of malicious URLs, but attackers often monitor these lists and update their domains/IPs accordingly. Finally, ad-blockers may provide inadvertent protection by blocking malvertisements, a common source of EK infections.

E. The State of Exploit Kits

Numerous high profile EK authors have been arrested in recent years. Most notably is the BlackHole author who is believed to have earned roughly $2.3 million in subscription fees [13]. When BlackHole was taken down in 2013, new EKs rose to prominence, and, by the end of 2015, Angler, Magnitude, Neutrino and Nuclear controlled 96% of the market [14].

By the end of 2016, when the developer behind the notorious Angler EK was arrested, the team behind Nuclear had already ceased its operations. Shortly after this, Neutrino was shut down following a joint operation, and researchers at Proofpoint noted that overall EK activity was down 93% [14]. This trend continued in 2017, showing EK activity falling a further 62% on 2016 [15]. Sundown was another EK to disappear at the end of 2017, following a source-code leak.

Experts believe several factors contributed to this decline; the fear of arrest, more profitable options for cyber-criminals, stronger offence by the security vendors, increased security in operating systems and browsers, the use of ad-blockers and more robust patching by software vendors [16]. However, the Rig EK remains active and has mostly dominated the market since 2017. The popularity of EKs may fluctuate, but as long as vulnerabilities exist in software, EKs remain a valid threat.

III. Related Work

Takata et al. [17] crawled 20,272 malicious websites over 4 years, extracting 8467 JS samples. They visited each website with a browser emulator (honeyclient), and, a real browser (targeted client) with different JS implementations. Two sets of HTTP entries are produced for each website which are then mapped into redirection graphs to identify any structural differences. This experiment led to the discovery of five previously unknown evasion techniques which exploit different JS implementations. REdiREKT shares some similarities; basic JS obfuscation methods are accounted for, and redirection chains are mapped. However, we also consider content-based redirects, and the goals of the two projects differ significantly.

Nikolaev et al. [18] presented a method of detecting EKs using features solely obtained from HTTP proxy logs, which are commonly available to organisations. The system was tested against HTTP logs from 200+ networks of various

sizes over 6 months, identifying hundreds of EKs with 99% precision. The fields extracted from HTTP proxy logs are also generated by Zeek and used by REdiREKT. However, their system aims to detect a single malicious flow (HTTP request/response pair) indicating that the website is an EK. It fails to consider the chain of redirections that led to the EK.

Suren et al. [19] applied ML to identify EK attacks using URL-based features, obtaining up to 100% accuracy. They used Zeek to process PCAPs collected from one of the same sources as REdiREKT [20], extracting 20 features for each URL in each stage of an EK attack. REdiREKT also uses URL-based features, but attackers can easily update the structure of URLs to bypass detection. Furthermore, the experiment only utilises 96 PCAPs from 2016, spanning 4 families. This limited dataset cannot accurately represent the EK ecosystem.

Harnmetta et al. [21] demonstrated a technique to classify EK behaviour via ML. The authors extract various content-based, interaction-specific and connection-specific features from the HTTP, DNS and Files logs produced by Zeek. They applied a decision tree classification model to the network flows extracted from the PCAPs and detected EK traffic and EK family types with over 97% accuracy. This experiment focuses on individual network flows rather than the full combination of flows that make up an EK attack.

Takata et al. [22] implemented a browser-emulator called MineSpider to extract malicious URLs. MineSpider applies program slicing to JS, executes each code segment and then extracts any resulting URLs, even when cloaking prevents malicious JS branches from being executed. When applied to over 19,000 malicious websites, MineSpider extracted more than 30,000 new URLs that had high levels of maliciousness. MineSpider fails to map redirection chains accurately because it only focuses on JS-based redirects. Referrer, Location, HTML and iFrame-based redirects should also be considered.

Takata et al. [23] proposed a system for detecting malicious websites by combining redirection and JS execution graphs. Like REdiREKT, the system processes referrer, location, HTML, iFrame and JS-based redirections. However, they also dynamically execute JS to uncover obfuscated content. When applied against a dataset of 2058 compromised websites from 2011-2015, the system was able to identify the precise position of compromised web content and the targeted client environment for 71.9% of the sites for which a redirection graph could be extracted. Failure to extract redirection graphs was typically a result of cloaking features used by EKs.

Stringhini et al. [24] implemented SpiderWeb which builds redirection graphs by aggregating the redirection chains from a collection of different users. SpiderWeb aims to detect all types of web-based malware, not just EKs. Some of the features extracted are also used by REdiREKT, e.g. chain length and TLD, but, by choosing not to differentiate between header and content-based redirects, SpiderWeb is unable to collect valuable data that can help to identify malicious behaviour.

Shibahara et al. [25] leveraged redirection subgraph similarities to identify evasive, malicious websites. Classification is performed using a graph mining approach, where the similarities between redirection subgraphs of malicious, benign and compromised websites are integrated. Using data from 455,860 crawled websites, they achieved a 91.7% true positive rate (TPR) for malicious sites hosting EK URLs with a 0.1% false positive rate (FPR). The system models redirections regardless of whether they occur, e.g. if a URL is found in JS but was not accessed, it is still labelled as a redirect. The purpose of this design choice is to bypass evasion methods, but it creates the potential for inaccurate results.

Matsunaka et al. [26] proposed a Framework for Countering Drive-by Download (FCDBD) which consists of monitoring sensors on the client-side (browser, web proxy and DNS), and, an analysis centre on the server-side. FCDBD identifies executable file downloads and classifies as malicious if the download URL is not present in any of the preceding HTTP headers or HTML/JS content. In our experience, it is not uncommon for EKs to populate header fields or store plain-text URLs in webpage content. Furthermore, URLs accessed within 2 seconds of each other are labelled as related but URLs accessed after user interaction, e.g. moving the mouse, are labelled as initial entry points (potentially incorrectly).

Mekky et al. [27] suggested reconstructing user browsing activity into trees to detect malicious redirections. Using data gathered from a large ISP, they extract header and content-based redirects, and, build trees consisting of nodes (domains) and edges (redirects). Supervised ML is performed using a decision tree classifier, achieving precision and recall values of up to 98%. The system shares some features used by REdiREKT and builds trees in a similar manner. However, it collects data via a browser add-on rather than network traffic and uses a private dataset that cannot be publicly verified.

Taylor et al. [28] developed a detection method which builds web session trees consisting of node and redirection-based features. After modelling some trees for known EKs and building a dataset of unclassified trees from 3800 hours of real-world traffic, the unclassified trees are compared to EK trees according to individual nodes and overall tree structure. Unlike REdiREKT, the system models web resources into trees rather than just the domains visited. Furthermore, content-based redirects are not considered, and the 5-second threshold used to define a session can be easily bypassed by attackers.

Nagai et al. [29] proposed building website structure trees (WS) to identify malicious websites without relying on complete information about redirections. The WS-trees model the structure of a website, e.g. paths and files are added as nodes. However, cross-domain redirects are considered. In our experience, EKs heavily utilise content-based redirections across multiple domains. Failing to account for these redirections leads to significant data loss. Furthermore, the small dataset of 256 websites misses several high profile EK families.

Nelm et al. [30] produced WebWitness, a system designed to investigate and categorise the web browsing paths followed by users before an attack occurs. WebWitness identifies a malicious download and then traces back through HTTP requests, building a tree of the redirections that led to the malware. The system extracts many of the same redirection-based features

as REdiREKT, but the goal is different; WebWitness aims to differentiate between DBD, social engineering and fake update attacks, rather than specifically focusing on EK detection.

MadTracer [31], WarningBird [32] and SURF [33] leverage redirection-based data for malicious website detection but have distinctly different goals to REdiREKT. MadTracer aims to detect malicious advertisements, WarningBird focuses on malicious URLs found on Twitter, and, SURF addresses SEO poisoning campaigns. Chen et al. [34] combined CSS and URL-based features to identify hidden malicious redirects. Using ML, they were able to achieve accuracy of up to 99%. However, attackers can circumvent via CSS obfuscation.

IV. Experiment

The experiment was conducted in September 2019 on a Windows 10 machine with an Intel i7-8700K CPU, 32GB RAM and a 350Mbps internet connection. The two virtual machines (VM) used (Ubuntu 18.04 and Windows 10) were assigned 8GB RAM and a 100GB HD each.

A. Goals

1) Build a ground-truth dataset of EK samples.
2) Develop a new system to generate a benign dataset.
3) Map HTTP redirection chains to identify the key differences between benign and malicious redirections.
4) Provide insights into the history of EKs and how the structure and methods of redirections have evolved.
5) Extract, combine and store redirection and HTTP-based features to assist detection-focused research.
6) Evaluate the system's performance and use-cases.

B. Methodology

REdiREKT utilises the Zeek IDS (formerly Bro) [35] to process HTTP traffic. By default, Zeek can read a PCAP file and generate a variety of useful logs. For this experiment, we are investigating features obtained from the HTTP log, but in future work, the DNS and files logs will be considered. Critically, Zeek has a custom scripting language which allows users to create and share plugins.

*1) **Extracting Redirections**:* Each sample is processed according to the following broad steps. The PCAP is read by Zeek and generates a HTTP log which consists of a series of HTTP entries (request/response pairs). These entries contain information such as the timestamp, source/destination IP, host, referrer, URI, user agent, method, response code etc. Zeek does not collect server headers (e.g. location) by default, but, a script can be integrated to ensure these headers are processed. The referrer/location headers can easily be used to map redirections because they cannot be obfuscated.

- **Referrer:** The `referrer` field is the redirect source, and the `host` field holds the destination URL.
- **Location:** The `host` field is the redirect source, and the `location` field holds the destination URL.

Although these headers account for a large portion of redirects, other redirection methods are more challenging to identify. Content-based redirections may occur via HTML, JS or iFrame, and, are commonly observed in malicious attacks as they help to obscure the redirection chain. This is especially true when the actual redirection code is obfuscated in some way. Although EKs use content-based redirects heavily, they are not inherently malicious and are used by benign websites. We created a Zeek script to extract content-based redirections.

The script reassembles HTTP bodies as Zeek processes them. Regular expressions are applied to the full HTTP bodies to extract various types of content-based redirections. Each unique redirect is stored as a log entry consisting of the timestamp, UID, redirect source and destination URL, and, the type of redirection that occurred. These potential redirects will be subject to a thorough validation process later. The regular expressions are designed to defeat simple obfuscation techniques such as whitespace randomisation, case sensitivity and foreign characters. Content-based redirects supported:

- **HTML:** Commonly observed HTML redirection methods, e.g. `http-equiv="Refresh" url="<url>"` and `form|a|p|img src="<url>"`.
- **JavaScript:** The contents of all `<script>` tags and JS files. Known redirect methods are also identified, e.g. `window|document(.location|.open)?.href|hostname|replace|assign|write`.
- **iFrame:** The contents all iFrame tags, e.g. `<iframe src="<url>"></iframe>`.
- **Base64:** Potential Base64 encoded strings are decoded, e.g. `window.cback('aHR0cDovL2V2aWwuY29tL2V4cGxvaXQvZXhwbG9pdC5waHA=');`.
- **Concatenation:** Suspected split-strings are joined, e.g. `var a = "http://" + 'evil' + ".com"; window.href=a;`.
- **Unknown:** The HTTP body is scanned for any remaining URLs, these redirects are only considered if the URL was visited and no other source was identified.

*2) **Building Redirection Trees**:* The HTTP entries from a processed PCAP are split into sets according to the source IP, this allows the tracking of multiple hosts. The content-based redirections are then mapped to the HTTP entries in each set. This process involves identifying a domain in the HTTP entries which matches the redirect destination URL and ensuring that the site was visited after the source URL, e.g. if an iFrame is found on a.com which points to b.com, we verify that b.com was accessed after a.com. If the redirect is deemed viable, the timestamp is updated to reflect the time of destination URL access and the content-based redirect is stored alongside the appropriate HTTP entry, providing that no duplicates exist.

The Python AnyTree module [36] is used to model redirections as trees, which comprise of nodes (domains) and edges (redirects). Trees may contain multiple redirection chains, which, we define as a path from the root node (entry domain) to a leaf node (exit domain). Each node holds a record of the domain visited, the type of redirects that occurred and the time between redirects. The set of HTTP entries for each source IP are further broken into temporal sessions; if the time difference between two neighbouring entries is greater than 15 minutes,

the preceding entries are split into a new session.

Each session is passed to a recursive algorithm to build a tree, modelling the header and content-based redirects for each entry. It is possible to have multiple redirection types for the same pair of nodes, e.g. referrer and iFrame. When all trees have been constructed, any single node trees (root node with no children) that exist are compared against the leaf nodes of each tree for a final redirection type; subdomain. This feature is similar to the "same-domain" feature in [30] and is based on the same observation; EKs often serve different exploits and payloads from alternate subdomains within the same domain.

- **Sub-Domain:** If a leaf node and single-node tree have different subdomains but matching domains, and, they were accessed within 60 seconds of each other.

*3) **Extracting Redirection Chains**:* Once all trees have been returned, the chains are extracted. Each redirection chain is a unique path from a root node to one of its leaf nodes. Critically, any of the root node's children that are not part of the direct path to the leaf node are removed. This is based on the understanding that in an EK attack chain, the root node represents the compromised host and most of the redirections from this node are benign. After the malicious code injected into the compromised site spawns a new redirection, we can assume all subsequent redirections on this path are malicious, as these are domains owned and controlled by the attackers.

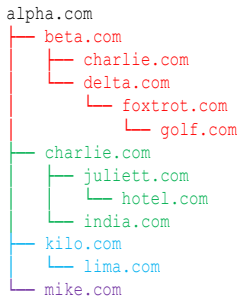

Fig. 1. Redirection Tree

Figure 1 provides a visual example of a tree and the four chains which are extracted from it. Note we first extract the chain from alpha.com to golf.com without modelling the siblings of beta.com (children of alpha.com which are not on the path to golf.com). However, we include charlie.com in the chain, despite it not being on the direct path from alpha.com to golf.com. This is because, in an attack, beta.com would be an attacker-controlled domain and all redirects should be modelled, regardless of whether they lead directly to the EK.

*4) **Feature Extraction and Storage**:* When processing the benign dataset, the redirection chains for each sample are extracted and stored as JSON objects, which, can accurately represent the tree-like structure. For the malicious dataset, all chains are extracted, but only the malicious chain is stored. To accomplish this, each chain is compared against its corresponding sample entry in a CSV file which holds a verified record of the compromised host and EK domain. If a match is found, the chain is extracted and tested against the known correct chain stored in a JSON file. This verification and testing procedure ensures quality is maintained throughout the project; the effects of any changes are easily observable and any bugs introduced are quickly identified.

Finally, the verified chains are used to extract 48 features for a ML dataset. The features are stored in CSV format using Python Pandas, allowing for efficient ML integration. Table I shows the features extracted for each node in a chain, some of which represent two features (average and total). The range of features is likely to evolve as the data provides insights and new sources are explored, e.g. DNS, file-based.

TABLE I
NODE-BASED FEATURES

Type	Feature	Description
HTTP	Requests	No of HTTP requests
HTTP	Response Length	Avg/Total length of HTTP responses
Redirect	Number	Index of node within chain
Redirect	Depth	Depth of node within chain
Redirect	Time	Time between redirections
Redirect	Referrer	Presence of 'Referrer' redirect
Redirect	Location	Presence of 'Location' redirect
Redirect	HTML	Presence of 'HTML' redirect
Redirect	JS	Presence of 'JS' redirect
Redirect	iFrame	Presence of 'iFrame' redirect
Redirect	Subdomain	Presence of 'Subdomain' redirect
Redirect	Concatenation	Presence of 'Concat' redirect
Redirect	Base64	Presence of 'Base64' redirect
Redirect	Unknown	Presence of 'Unknown' redirect
URL	Standard Port	Domain uses default HTTP(S) port
URL	Is IP	Domain is an IP address
URL	Domain Length	Length of the domain name
URL	Domain Entropy	Entropy of the domain name
URL	URI Length	Avg URI length
URL	URI Entropy	Avg URI entropy
URL	URI Slash	Avg/Total slashes ('/')
URL	URI Amp	Avg/Total ampersands ('&')
URL	URI Dash	Avg/Total dashes ('-')
URL	URI Plus	Avg/Total pluses ('+')
URL	TLD	Top-level domain (cat-encoded)
Content	Bytes Shockwave	Avg/Total Shockwave bytes
Content	Bytes Executable	Avg/Total EXE bytes
Content	Bytes Java	Avg/Total Java bytes
Content	Bytes Silverlight	Avg/Total Silverlight bytes
Content	Bytes JavaScript	Avg/Total JavaScript bytes
Content	Bytes XML	Avg/Total XML bytes
Content	Bytes ZIP	Avg/Total ZIP bytes
Content	Bytes Image	Avg/Total Image bytes
Content	Bytes HTML	Avg/Total HTML bytes

Figure 2 provides a high-level overview of how each PCAP is processed by REdiREKT.

C. Data Collection

The data for this experiment was carefully collected and verified to produce a ground-truth dataset for future research.

*1) **Malicious**:* The malicious dataset was collected from two reputable sources; malware-traffic-analysis.com [20] and broadanalysis.com [37]. These websites have hosted malware samples since 2013 and 2016 (respectively), many of which are labelled as EK traffic. Each PCAP is accompanied by a blog post containing a detailed analysis of the sample. These reports allow essential information about EK traffic to be quickly identified e.g. family, campaign, exploit/malware type and associated domains. Only PCAPs labelled as EK-traffic containing at least one redirection were considered. Sometimes

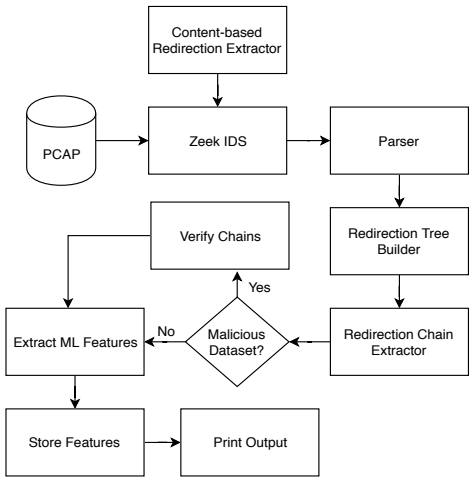

Fig. 2. REdiREKT High-Level Flowchart

this could not be determined from the sample name, so each blog entry was carefully reviewed. Some of the PCAPs were not suitable for the experiment and were excluded:

- Samples with only post-infection traffic were discarded.
- Spam-based EK attacks were not processed. Some malvertising samples were stored as the redirection structure closely resembles compromised website-based EK attacks. These samples were clearly labelled for easy exclusion from statistics and the ML dataset.
- Zeek could not process some PCAPs due to corruption, often resulting from confidential data or noise being pre-removed from the sample. A small modification of the Zeek core code allowed many of these corrupted samples to be processed but some ultimately had to be discarded.
- A small number of ZIP archives containing PCAP samples could not be opened due to an incorrect password which the original author was unable to provide.

Many samples were missing some of the initial traffic, e.g. the compromised host, often due to the data being stripped to protect privacy. We did not discard these PCAPs, providing they contained at least one redirection. However, any samples missing the compromised host were clearly labelled, allowing us to accurately update statistics and exclude the samples from ML if required. After the data collection process, we were left with 1279 malicious samples from 2013-2019, spanning 28 EK families and 8 campaigns. Figure 3 contains a breakdown of the most popular EK families; the 'Other' category consists of 18 families which each had fewer than 10 samples.

We compiled a CSV file containing the URL of the compromised host and EK for each sample according to the values found in the blog entries. Next, we carefully processed each sample with REdiREKT, comparing the root and leaf nodes of each redirection chain against the corresponding entry in the CSV file. If a chain beginning and ending with the correct compromised host and EK domain is not found, the sample is manually analysed to determine the cause. If the issue cannot be resolved, e.g. advanced obfuscation is preventing us from linking the compromised host and EK; the sample is labelled

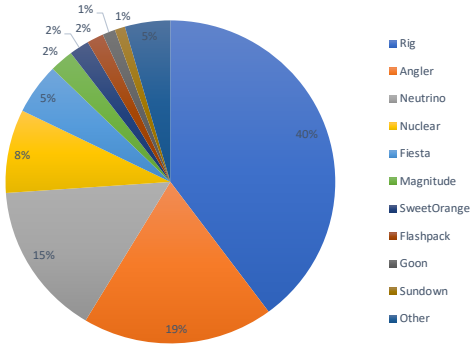

Fig. 3. EK Family Breakdown

as incorrect, and the obfuscated code is stored for future analysis. A JSON object with the correct redirection chain is manually compiled. This ensures statistics are correctly captured and provides a test case to re-validate results.

If a chain beginning and ending with the correct compromised host and EK domain is extracted; each node in the chain is compared against the corresponding blog entry to ensure that no redirections were missed and the nodes are ordered correctly. If this is the case, the sample is labelled as correct and the chain is exported to a JSON object. If there are any discrepancies in the results, the sample is manually analysed to identify the problem. If the issue cannot be resolved, the sample is labelled as semi-correct, and the JSON object is manually corrected to serve as a test case. Note that although the naming convention of the samples was generally consistent, many required updating to ensure that family/campaign-based statistics could be accurately captured. Furthermore, sample names were updated to reflect the type of EK attack and indicate whether data was missing.

*2) **Benign**:* The benign dataset was generated from the Alexa top 10k websites. First, we used Google BigQuery to obtain a list of the top 1 million websites, as indexed by Censys [38]. We selected the top 10k domains and pre-filtered them using the Scrapy framework. This allowed us to quickly filter out any duplicates, domains that fail to load and HTTPS-based PCAPs, which cannot be decrypted by Zeek (see Section V). The pre-filtering process produced 1525 unique domains. Although this number may appear low, over 70% of the top 10k domains use HTTPS, which, is to be expected for high-profile websites. The remaining domains failed due to a variety of errors, e.g. connection refused, DNS lookup and timeout.

To generate PCAPs for the 1525 extracted domains, Selenium is used to visit each domain and traffic is captured via TShark. The system operates on Windows 10 using the native browser to recreate the machines used to capture the malicious traffic as closely as possible. First, each domain is queried using the VirusTotal API [39]. If any AV vendors flag the URL as malicious, it is excluded from further processing. 88/1525 of the domains were excluded for this reason.

If the domain was flagged as benign, the browser opens the URL and waits 60 seconds for the page to load. If the page times out, the domain is excluded from the dataset. 37/1525 domains were excluded for this reason. If the page loads

correctly, the Selenium driver attempts to close any generic GDPR/cookie-related pop-up windows and continues to capture traffic for 15 seconds. The packet capture is subsequently terminated, and all browser windows are closed, ensuring a new session for each sample. This system produced 1400 PCAPs which were then processed through REdiREKT.

D. Results

*1) **Malicious vs Benign**:* 3328 (96.52%) malicious domains were correctly extracted from 1279 malicious chains. 1172 (91.63%) malicious chains were correctly identified, 98 (7.66%) were semi-correctly identified, and 9 (0.7%) were incorrectly identified. Note, semi-correct chains are chains beginning and ending with the correct domain, but, one or more redirections are missing or ordered incorrectly. The compromised host was missing for 127 (9.93%) of the malicious PCAPs, and 53 (4.14%) were labelled as malvertising.

12,783 benign domains were extracted from 5910 benign chains. 1258 (89.9%) PCAPs were processed successfully. The remaining were either empty or produced parsing errors in Zeek. 581 (40.79%) of the successful PCAPs contained zero redirections. A breakdown of redirection statistics for the malicious and benign dataset are presented in Table II.

TABLE II
MALICIOUS VS BENIGN REDIRECTIONS

Redirections	Malicious	Benign
Referrer	1381 (38.88%)	5804 (67.72%)
Location	194 (5.46%)	379 (4.42%)
HTML	337 (9.49%)	908 (10.59%)
JavaScript	729 (20.52%)	893 (10.42%)
iFrame	583 (16.41%)	152 (1.77%)
Base64	51 (1.44%)	0
Concat	14 (0.39%)	0
Subdomain	6 (0.17%)	52 (0.61%)
Advanced	120 (3.38%)	0
Unknown	137 (3.86%)	383 (4.47%)
Max Node Depth	8	7
Average Node Depth	1.67	1.05
Max Redirects	10	40
Average Redirects	2.7	2.16
Obfuscated Redirects	328 (9.23%)	435 (5.08%)
Average Redirect Time	4.09 (sec)	0.89 (sec)

The breakdown of redirections shows that the referrer header is far more common in benign chains while JS and iFrame are often used in malicious chains. Obfuscated redirects such as base64 and concatenation were not witnessed in any benign chains. The time between redirects varies significantly between the datasets, this may be due to numerous factors, e.g. the environment used to capture data, time period of data collection, and, the features of the website being tested.

*2) **Malicious - Yearly**:* We analysed the yearly trends of the malicious results to gain insight into the evolution of EKs. The number of samples available from year to year varies, e.g. 2013, 2018 and 2019 have 15-25 samples each while 2014-2017 comprises of 1222 samples. This must be taken into consideration when assessing the accuracy of the results. The lack of recent samples is mostly a result of the decline of exploit kit popularity [40], but these trends can quickly change. The yearly detection results are shown in Table III, containing the percentage of chains correctly identified and the number of malicious URLs correctly extracted. A breakdown of the yearly redirection types is displayed in Figure 4.

TABLE III
MALICIOUS YEARLY RESULTS

Year	PCAPs	Correct	Semi-Correct	Incorrect	URLs
2013	15	93.33%	0%	6.67%	94.55%
2014	246	93.5%	6.5%	0%	98.03%
2015	162	74.69%	25.31%	0%	91.16%
2016	677	95.13%	4.73%	0%	97.97%
2017	137	95.62%	4.38%	0%	98.18%
2018	19	84.21%	0%	15.79%	85.45%
2019	23	69.57%	13.04%	17.39%	78.18%

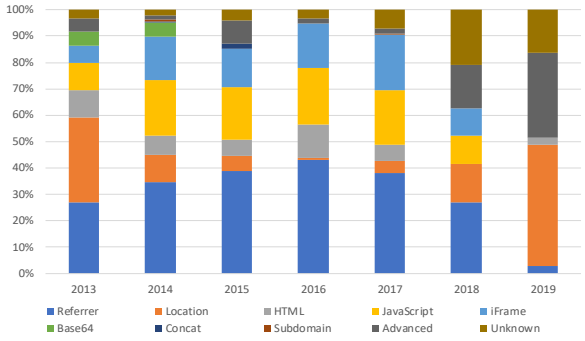

Fig. 4. Malicious Yearly Redirections

These results appear to indicate that obfuscation techniques increased in sophistication during 2018-2019, where we observe more advanced redirections and fail to correctly identify a greater proportion of malicious chains than previous years. However, this could also be due to a lack of available data for this period. There are a high number of semi-correct chains identified in 2015; this was due to the high frequency of samples from specific campaigns which used advanced obfuscation techniques. For example, 38/162 of the PCAPs from 2015 were part of the BizCN campaign which combined the use of unicode and concatenation-based obfuscation to hide redirects. It would be trivial to detect these examples, but it is desirable to avoid de-obfuscation methods that are too family-specific. Although only 74.69% of the chains extracted from 2015 were correct, 91.16% of URLs were correctly extracted, which correlates with the rest of the 2013-2017 results.

*3) **Malicious - Families**:* We compared the results for each EK family and found that referrer was the most common redirection method, except for Fiesta. JS, HTML and iFrame typically make it into the top 5 redirections. This doesn't include Nuclear or Fiesta where HTML only accounted for 5.9% and 0% of redirections, respectively. The location header is an outlier for the Nuclear EK, accounting for 10.8% of redirections compared to the average of 5-6% observed in typical benign and EK traffic. Nuclear also has a high number of advanced redirections; this is due to the high presence of samples from the BizCN gate campaign discussed previously. Table IV shows the top 5 redirection types for the top 5 EK's.

*4) **Performance**:* The malicious dataset of 1279 PCAPs was processed in 29 minutes and 4 seconds, with an average

TABLE IV
MALICIOUS FAMILY REDIRECTIONS

Rig	Angler	Neutrino	Nuclear	Fiesta
Referrer (39.7%)	Referrer (46.8%)	Referrer (35.5%)	Referrer (39.8%)	JavaScript (37.6%)
JavaScript (23.4%)	JavaScript (17.5%)	iFrame (21.2%)	JavaScript (18.1%)	Referrer (33.8%)
iFrame (18.6%)	HTML (15.2%)	JavaScript (18.8%)	Location (10.8%)	iFrame (20.2%)
HTML (8.6%)	iFrame (10.5%)	HTML (14.3%)	iFrame (10.5%)	Concat (6.6%)
Location (3.4%)	Unknown (5.2%)	Unknown (4.5%)	Advanced (10.2%)	Location (1.4%)

of 1.36 seconds. The benign dataset of 1400 PCAPs was processed in 20 minutes and 53 seconds, with an average of 0.9 seconds. Note, the functionality used to detect concatenation-based redirections is resource-intensive and if removed, results in a 3x speed boost. Considering it only accounts for 0.39% of malicious redirects and 0% of benign redirects, it may be beneficial to discard it for cases that require high performance, e.g. if the system is applied against real-time network traffic.

E. Evaluation

The annual malicious data appears to indicate an increase in advanced obfuscation techniques in recent years. Although there was not enough data to prove this conclusively, other researchers have made similar observations [41]. The failure to identify heavily obfuscated redirections is the primary flaw of REdiREKT, but this weakness is shared with many other conventional systems as script de-obfuscation remains an open research problem [42], [43]. Attackers could circumvent detection by using HTTPS, this would render the data unreadable by Zeek, and it would not be possible to extract key features for ML. However, only 0.18% of the malicious PCAPs used HTTPS, and there are methods to analyse malicious HTTPS traffic should it become a common trend (see Section V).

Attackers could mix benign redirects into the EK chain to try and bypass detection, and poison ML models [12] as REdiREKT classifies any redirections spawned from an attacker-controlled domain as malicious. However, an attacker cannot insert benign redirects directly between two EK-related domains; they don't control the benign domain and would be unable to force a further redirection to their EK. They could redirect the victim to benign domains to add noise while keeping the original EK chain intact, but forwarding victims from an EK domain to a benign domain would increase the chance of detection as the benign website admin may investigate the traffic or receive alerts from confused victims.

When capturing traffic using the Selenium instrumented browser, we wait 60 seconds for the page to load and then a further 15 seconds for additional traffic. The attacker could bypass this by introducing a delay, so the initial redirect would not be captured. This would undoubtedly lead to a lower infection rate for attackers as they would fail to exploit any users who leave the page within 15 seconds of it loading. Therefore, this technique is unlikely to gain traction, and it could be countered by increasing the Selenium wait-time.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we presented REdiREKT, a system designed to map HTTP redirection chains and extract distinguishing features. We tested REdiREKT against 1279, manually verified malicious samples and successfully extracted 96.52% of malicious domains while only failing to extract 0.7% of malicious chains. We also built a dataset of 1400 benign samples from the Alexa top 10k websites and extracted 12,783 domains from 5910 redirection chains. Using features identified from existing research and discovered from our analysis of EKs, we compiled a database of ML features to aid the development of new detection techniques. To our knowledge, our experiment provides the first set of entirely accurate EK redirection statistics. This is because we used manually compiled and verified redirection objects, rather than relying on the results of REdiREKT which did not achieve 100% accuracy.

One of the main goals of this project was to collect a dataset of features for ML, so this is a primary focus area for future research. Some existing research use similar features to those currently collected by REdiREKT [18], [21] but these experiments typically focus on identifying a malicious URL rather than a malicious chain. Those which do consider redirections, only extract node-based features from the domain which delivers the exploit/malware. We conclude that only considering the features of the domain which delivers the exploit leads to the loss of important data. Research focusing on the malicious redirection chains [17], [24]–[30] have slightly different goals or methods, as described in section III.

Another area of future work will involve expanding the range of features considered for ML. The DNS and file-based features are currently captured by Zeek. We could identify the age and scheduled validity of domain names as a feature, e.g. if a domain was registered recently and has a short expiration, it's more likely to be malicious. Similarly, if an executable file is downloaded by the final node in a chain, it is more likely to be an EK attack than a chain that does not drop a binary. Additional de-obfuscation methods could also be implemented to increase accuracy, providing they aren't too family-specific.

During the experiment, we found that over 70% of the Alexa top 10k websites use HTTPS whilst 0.18% of EK samples use HTTPS. Future work could involve intercepting live traffic, stripping HTTPS and forwarding the decrypted traffic to Zeek. This would be relatively trivial but is only suitable if REdiREKT can be placed at the edge of the network, with authorisation to intercept and scan traffic. Expanding the dataset is another goal, but this is complicated by the dynamic nature of EK activity. There are additional repositories which host EK PCAPs, but first, these samples must be manually verified and labelled, which is a time-consuming process.

REdiREKT could introduce potential performance overhead to networks with high levels of traffic, and, this computational cost is expected to rise as machine learning detection methods

are applied. It is therefore envisioned that the system could be offered as a cloud-based solution. Customers who sign up for this cloud service may also be incentivised to share anonymised data in order to monitor and improve the effectiveness of the ML classifier and REdiREKT overall.

REFERENCES

[1] Symantec. (2019) Internet security threat report volume 24. [Online]. Available: https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf

[2] J. Burgess, D. Carlin, and P. O'Kane, "Manic: Multi-step assessment for crypto-miners," in *2019 International Conference on Cyber Security and Protection of Digital Services*. IEEE, 2019, pp. 1–8.

[3] G. De Maio, A. Kapravelos, Y. Shoshitaishvili, C. Kruegel, and G. Vigna, "Pexy: The other side of exploit kits," in *International conference on detection of intrusions and malware, and vulnerability assessment*. Springer, 2014, pp. 132–151.

[4] B. Eshete and V. Venkatakrishnan, "Webwinnow: Leveraging exploit kit workflows to detect malicious urls," in *Proceedings of the 4th ACM conference on Data and application security and privacy*. ACM, 2014, pp. 305–312.

[5] H. Li, S. Maffeis, and M. Cheraghchi, "Amj: An analyzer for malicious javascript," 2018.

[6] N. Biasini. (2015) Threat spotlight: Cisco talos thwarts access to massive international exploit kit generating 60m annually from ransomware alone. [Online]. Available: https://www.talosintelligence.com/angler-exposed

[7] J. E. Nick Biasini. (2015) Threat spotlight: Angler lurking in the domain shadows. [Online]. Available: https://blogs.cisco.com/security/talos/angler-domain-shadowing

[8] V. Kotov and F. Massacci, "Anatomy of exploit kits," in *International symposium on engineering secure software and systems*. Springer, 2013, pp. 181–196.

[9] Y. Shiraishi, M. Kamizono, M. Hirotomo, and M. Mohri, "Multi-environment analysis system for evaluating the impact of malicious web sites changing their behavior," *IEICE TRANSACTIONS on Information and Systems*, vol. 100, no. 10, pp. 2449–2457, 2017.

[10] T. Taylor, K. Z. Snow, N. Otterness, and F. Monrose, "Cache, trigger, impersonate: Enabling context-sensitive honeyclient analysis on-the-wire," in *NDSS*, 2016.

[11] G. Chris, L. Ballard, J. Caballero, N. Chachra, C. J. Dietrich, K. Levchenko, and P. Mavrommatis, "Manufacturing compromise: the emergence of exploit-as-a-service," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012.

[12] E. Suren and P. Angin, "Know your ek: A content and workflow analysis approach for exploit kits." *J. Internet Serv. Inf. Secur.*, vol. 9, no. 1, pp. 24–47, 2019.

[13] B. Krebs. (2016) Blackhole exploit kit author gets 7 years. [Online]. Available: https://krebsonsecurity.com/2016/04/blackhole-exploit-kit-author-gets-8-years/

[14] T. Spring. (2017) Where have all the eks gone? [Online]. Available: https://threatpost.com/where-have-all-the-exploit-kits-gone/124241/

[15] D. S. A. Team. (2017) Fluctuation in the exploit kit market – temporary blip or long-term trend? [Online]. Available: https://www.digitalshadows.com/blog-and-research/fluctuation-in-the-exploit-kit-market-temporary-blip-or-long-term-trend/

[16] D. Kaplan. (2017) Why exploit kits are going dark? [Online]. Available: https://www.trustwave.com/Resources/Trustwave-Blog/Why-Exploit-Kits-Are-Going-Dark/

[17] Y. Takata, M. Akiyama, T. Yagi, T. Hariu, K. Ohkubo, and S. Goto, "Identifying evasive code in malicious websites by analyzing redirection differences," *IEICE Transactions on Information and Systems*, vol. 101, no. 11, pp. 2600–2611, 2018.

[18] I. Nikolaev, M. Grill, and V. Valeros, "Exploit kit website detection using http proxy logs," in *Proceedings of the Fifth International Conference on Network, Communication and Computing*. ACM, 2016, pp. 120–125.

[19] E. SÜREN, P. ANGIN, and N. BAYKAL, "I see ek: A lightweight technique to reveal exploit kit family by overall url patterns of infection chains," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 27, no. 5, pp. 3713–3728, 2019.

[20] B. Duncan. (2020) Malware traffic analysis. [Online]. Available: https://www.malware-traffic-analysis.net/

[21] S. Harnmetta and S. Ngamsuriyaroj, "Classification of exploit-kit behaviors via machine learning approach," in *2018 20th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2018, pp. 468–473.

[22] Y. Takata, M. Akiyama, T. Yagi, T. Hariu, and S. Goto, "Minespider: Extracting urls from environment-dependent drive-by download attacks," in *2015 IEEE 39th Annual Computer Software and Applications Conference*, vol. 2. IEEE, 2015, pp. 444–449.

[23] Y. Takata, M. Akiyama, T. Yagi, T. Yada, and S. Goto, "Fine-grained analysis of compromised websites with redirection graphs and javascript traces," *IEICE TRANSACTIONS on Information and Systems*, vol. 100, no. 8, pp. 1714–1728, 2017.

[24] G. Stringhini, C. Kruegel, and G. Vigna, "Shady paths: Leveraging surfing crowds to detect malicious web pages," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 133–144.

[25] T. Shibahara, Y. Takata, M. Akiyama, T. Yagi, K. Hato, and M. Murata, "Evasive malicious website detection by leveraging redirection subgraph similarities," *IEICE TRANSACTIONS on Information and Systems*, vol. 102, no. 3, pp. 430–443, 2019.

[26] T. Matsunaka, A. Kubota, and T. Kasama, "An approach to detect drive-by download by observing the web page transition behaviors," in *9th Asia Joint Conference on Information Security*. IEEE, 2014, pp. 19–25.

[27] H. Mekky, R. Torres, Z.-L. Zhang, S. Saha, and A. Nucci, "Detecting malicious http redirections using trees of user browsing activity," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 1159–1167.

[28] T. Taylor, X. Hu, T. Wang, J. Jang, M. P. Stoecklin, F. Monrose, and R. Sailer, "Detecting malicious exploit kits using tree-based similarity searches," in *proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*. ACM, 2016, pp. 255–266.

[29] T. NAGAI, M. KAMIZONO, Y. SHIRAISHI, K. XIA, M. MOHRI, Y. TAKANO, and M. MORII, "A malicious web site identification technique using web structure clustering," *IEICE TRANSACTIONS on Information and Systems*, vol. 102, no. 9, pp. 1665–1672, 2019.

[30] T. Nelms, R. Perdisci, M. Antonakakis, and M. Ahamad, "Webwitness: Investigating, categorizing, and mitigating malware download paths," in *24th {USENIX} Security Symposium 15*, 2015, pp. 1025–1040.

[31] 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*. ACM, 2012, pp. 674–686.

[32] S. Lee and J. Kim, "Warningbird: Detecting suspicious urls in twitter stream." in *Ndss*, vol. 12, 2012, pp. 1–13.

[33] L. Lu, R. Perdisci, and W. Lee, "Surf: detecting and measuring search poisoning," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 467–476.

[34] B. Chen and Y. Shi, "Malicious hidden redirect attack web page detection based on css features," in *2018 IEEE 4th ICCC*. IEEE, 2018, pp. 1155–1159.

[35] Zeek. (2020) The zeek network security monitor. [Online]. Available: https://www.zeek.org/

[36] c0fec0de. (2020) Python anytree module. [Online]. Available: https://anytree.readthedocs.io/en/latest/

[37] B. Analysis. (2020) Broad analysis. [Online]. Available: https://broadanalysis.com/

[38] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by internet-wide scanning," in *Proceedings of the 22nd ACM SIGSAC Conference on CCS*. ACM, 2015, pp. 542–553.

[39] VirusTotal. (2019) Virustotal. [Online]. Available: https://www.virustotal.com

[40] Z. Ma, "The decline of exploit kits as an exploitation strategy," 2018. [Online]. Available: https://www.doc.ic.ac.uk/~livshits/papers/theses/zicong_ma.pdf

[41] T. Luo and X. Jin, "Next generation of exploit kit detection by building simulated obfuscators," in *Black hat conference*, 2016.

[42] W. Xu, F. Zhang, and S. Zhu, "Jstill: mostly static detection of obfuscated malicious javascript code," in *Proceedings of the third ACM conference on DASP*. ACM, 2013, pp. 117–128.

[43] P. Skolka, C.-A. Staicu, and M. Pradel, "Anything to hide? studying minified and obfuscated code in the web," in *The World Wide Web Conference*. ACM, 2019, pp. 1735–1746.