Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 3713 – 3728

doi:10.3906/elk-1810-199

I see EK: A lightweight technique to reveal exploit kit family by overall URL patterns of infection chains

Emre SÜREN[1*], **Pelin ANGIN**[2], **Nazife BAYKAL**[1]
[1]Department of Information Systems, Informatics Institute, Middle East Technical University, Ankara, Turkey
[2]Department of Computer Engineering, Faculty of Engineering, Middle East Technical University, Ankara, Turkey

Received: 27.10.2018 • **Accepted/Published Online:** 05.06.2019 • **Final Version:** 18.09.2019

Abstract: The prevalence and nonstop evolving technical sophistication of exploit kits (EKs) is one of the most challenging shifts in the modern cybercrime landscape. Over the last few years, malware infections via drive-by download attacks have been orchestrated with EK infrastructures. Malicious advertisements and compromised websites redirect victim browsers to web-based EK families that are assembled to exploit client-side vulnerabilities and finally deliver evil payloads. A key observation is that while the webpage contents have drastic differences between distinct intrusions executed through the same EK, the patterns in URL addresses stay similar. This is due to the fact that autogenerated URLs by EK platforms follow specific templates. This practice in use enables the development of an efficient system that is capable of classifying the responsible EK instances. This paper proposes novel URL features and a new technique to quickly categorize EK families with high accuracy using machine learning algorithms. Rather than analyzing each URL individually, the proposed overall URL patterns approach examines all URLs associated with an EK infection automatically. The method has been evaluated with a popular and publicly available dataset that contains 240 different real-world infection cases involving over 2250 URLs, the incidents being linked with the 4 major EK flavors that occurred throughout the year 2016. The system achieves up to 100% classification accuracy with the tested estimators.

Key words: Exploit kit, web malware, drive-by download, URL analysis, supervised machine learning, cybercrime

1. Introduction

Cyberattacks have been threatening Web visitors ever more with the widespread use of the Internet and exploit kits (EKs) have become one of the most disruptive weapons for Internet crimes. The emergence and prevalent use of EK infrastructures is one of the most dangerous developments in the cybercrime space according to one report.[1] EKs exhibit the current state-of-the-art crimeware that is capable of running in an automated fashion, achieving large-scale infection, and providing remote access. Therefore, the EK phenomenon is among the principal concerns of many security researchers and practitioners today.

In recent years, EKs have been progressively utilized for system compromise and malware propagation. These serve various types of malicious content via spear-phishing and drive-by download attacks, in which a payload is executed on user systems after a client-side vulnerability is exploited [1]. The drive-by download technique has had dramatic advancements in the past couple of years. Previously, malicious webpages were generated quickly in a simple manner. Then they evolved into frameworks, and today sophisticated attack tools

*Correspondence: emre.suren@metu.edu.tr

[1]https://blog.malwarebytes.com/cybercrime/2013/02/tools-of-the-trade-exploit-kits/

known as EKs are on the scene. EK mechanisms automate the infiltration process and command and control facility of a massive number of vulnerable machines and today they have become responsible for the majority of client-side attacks affecting Web visitors. The most common application on Internet-enabled devices is Web browsers, which are hot targets for EKs to infect the victim's system with a malware, and after exploiting a vulnerability, hackers usually steal information (e.g., credit card numbers) to directly use or encrypt private data of the user (e.g., text documents), then ask for a ransom to enable the decryption routine. Even worse, the compromised devices can become slaves leveraged to attack other systems without any notice. While the primary kind of attack launched through EKs is drive-by download, click-fraud (AdFraud) and cryptocurrency-mining are also hot alternatives.

The illustration in Figure 1 is a high-level overview of attacks based on an EK structure that contains 5 essential steps. Attackers utilize three major threat vectors for large-scale malware distribution, which are compromised webpages, malicious advertising (malvertisement), and malicious spam (malspam). This is known as a campaign and victims are drawn towards EKs by campaigns. Particularly, today the greater part of campaigns leverage compromised webpages to direct the target systems to an EK. Social networks and search term poisoning techniques are still highly utilized to quickly disseminate the infecting URLs throughout the Internet. There is an additional layer between campaigns and EKs known as a gate or traffic redirection system (TDS), which is deployed to transfer victims from campaigns to EKs. According to the victim profile, the EK infects the target system with a proper malware.

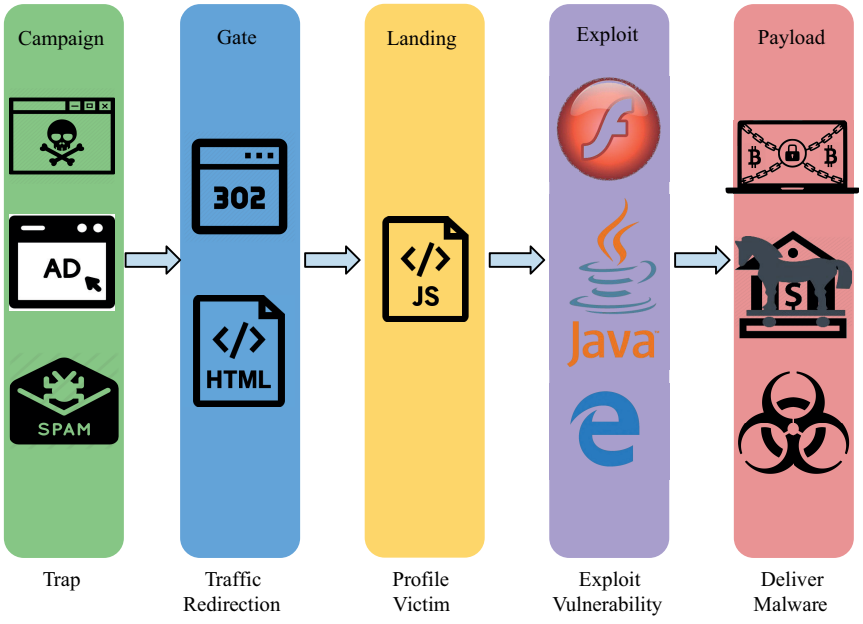

Figure 1. Five phases of EK infection chains.

A great deal of security research in the past decade has been dedicated to detecting standalone pieces of malicious code. The high number of infection cases and the high rate of changes in the malicious webpage ecosystem urged security practitioners to develop automated analysis systems, known as honeyclients [2–7]. These visit webpages and analyze the behaviors to detect the malicious ones. High-interaction honeyclients are instrumented virtual machines that contain real Web browsers. They visit webpages and subsequently collect artifacts on the operating system. In the case of exploitation, the instrumentation software notices newly

spawned processes, opened network connections, manipulated files, or registry entries and thus detects the attack. The output of the inspection is usually the blacklisted URLs and IoCs that are fulfilled by host-based IDS/IPS or EDR technologies. The critical point about high-interaction systems is that understanding whether there is an infection or not is relatively easier, possibly even for zero-day, but realizing the origin of the attack (e.g., attack platform and intrusion techniques) is quite difficult. Low-interaction honeyclients are instrumented with a headless browser that are usually wget, curl, PhantomJS, HtmlUnit, Selenium, or a custom-implemented Web client. They retrieve webpages and subject them to static and dynamic analysis of the Web content. The output of examination is usually the signatures for both the URLs and Web content that will be usable by Web content filtering systems. The vital point about low-interaction systems is that, while relatively very quick analysis is provided, HTML parsers and JavaScript interpreters are not as capable as real web browsers and malicious code usually targets them to break execution (e.g., invalid HTML tag), and correspondingly analysis.

Today, organized cybercrime on the Web is propagating via EKs, which smoothly evade traditional analysis systems. IDS/IPS and Web gateway security vendors focus on the EK complication to keep their signature database up-to-date by analyzing such network traffics. First, they usually develop regular expressions to detect infecting contents or just blacklist the URLs. On the other hand, creating a new unique signature takes time and effort, since the signature has to be able to match all variants of the EK family while not blocking benign webpages. However, it is not possible to find examples of the new samples at the first try. Secondly, signature-based inspection technologies require extensive maintenance in order to keep up-to-date rules against even minor changes in EK families. Therefore, due to the excessive number of signatures, those systems are not convenient for frequently changing environments.

Nowadays, security research centers sporadically capture network packets consisting of exploit and malware by utilizing honeypot mechanisms for early intelligence purposes. In order to get notified about zero-day threats as soon as possible, they plant as many trap systems as possible, which results in a huge volume of network traffic for analysis. However, traditional systems are not suitable for large-scale analysis in a reasonable amount of time. That is why researchers favor machine learning for threat intelligence in these times.

Most previous academic work [8–11] focused on the server-side source code of the EK families and conducted static source code analysis mostly on PHP code. While the EK families they analyzed leaked behind the scenes, the current EK families that we analyzed are not leaked online yet. The latter debates [12, 13, 15] involve machine learning to detect EK traffics from webpage content that are behind the attacks; however, content inspection consumes too many resources and have high time complexity. In addition, although binary classification as malicious or benign still dominates the literature today, it falls short of providing efficient threat intelligence, since the severity of each EK (exploited vulnerability, distributed malware, etc.) is not the same. More precisely, not all EKs are prevalent at the same time and not every EK is the same in terms of sophistication and posed danger. Therefore, EK family categorization is inevitable for advanced threat intelligence and the proposed system should be able to identify changes in EK-based attacks efficiently.

The purpose of this research is to recognize the network traffic of the state-of-the-art EK families efficiently with honeypot traffic analysis to simplify the work of security analysts. This study comprises the design, development, and evaluation of an efficient and original categorization method based on machine learning techniques. Experiments with real-world incidents demonstrate that the proposed models are highly efficient in categorizing EK families. The assessment shows that the stable classifier, I see EK (IsEK), yields an accuracy rate of at least 91.6%. The produced system relies on URL analysis rather than content inspection where novel URL features were introduced.

Accurately categorizing similar HTTP activities that belong to prevalent EK families is an important task for a number of reasons. If the assignment process is executed regularly for particular intervals, the classes that have the highest number of incidents indicate the EK families becoming prevalent. This enables researchers to abandon studies on discontinued EKs. It is also known that signature-based techniques turn off the rules related to unused attacks in order to achieve better performance. In addition, tracking the new attack and evasion techniques utilized by the attackers as closely as possible brings invaluable adversarial understanding. In this way, protection systems could be tuned better to make Internet visitors safer.

The exceptional elements of our approach are primarily related to the data source we utilize: we engage a real data source rather than generating our own. The data corpus is publicly available and stored in network packet captures (pcap). The data collected in a period of one year in 2016. The collection consists of real-world infections from 4 prevalent live EK families. To the best of our knowledge, there is no publicly released research that analyzes the EK traffic that occurred throughout 2016. The network traffic of malware infections through EKs is captured by deliberately accessing the malicious Web sources with real systems rather than relying on honeyclients.

The rest of this paper is organized as follows. A discussion of the literature and challenges is provided in Section 2. A comprehensive technical explanation of the methodology including experimental design (e.g., sampling strategy), feature selection details, and implementation (e.g., cross-validation of the classifiers) appears in Section 3. In Section 4, the developed classifiers are evaluated and compared, and then analysis of the results is presented. Finally, the paper is concluded in Section 5 with future study opportunities.

2. Related work

The first studies on EKs focused on analysis of the source code of EK families, in which researchers installed EKs from sources to their lab environment for inspection. The dataset contained in each work is partially common, covering different sets of EKs and back then frequently prominent ones.

Grier et al. [8] conducted a study on the emergence of the "Exploit-as-a-Service" model for the drive-by download landscape. Their dataset contained 77,000 malicious URLs taken from Google Safe Browsing and a blacklist provider. According to their research results, in total, over 10,000 unique executable files were delivered and dynamic analysis of those binaries led to 32 families of malware. In addition, several prominent types of malware are delivered even by an individual EK.

Kotov and Massacci analyzed the source code of 30 (partly inactive) different EK types to understand major behaviors and operational skills [9]. The preliminary analysis indicated that the major functionalities of EKs are managing exploits, evading detection mechanisms, and command and control. The manual examination concluded that 82% of the EKs apply obfuscation techniques. A handful of well-known vulnerabilities are targeted rather than launching zero-day exploits or sophisticated attacks.

Allodi et al. performed experiments via MalwareLab with the source code of 10 EKs to reveal the resilience to changes of targeted systems, particularly the operating system, browser, and plug-ins [10]. They deployed EKs in a controlled sandbox environment and found that some EK frameworks support the latest exploits, where cybercriminals achieve a higher infection rate in a small amount of time at the expense of short appearance on the market. On the other hand, some EK families prefer to serve more stable exploits, where attackers get a lower but steadier infection pace over time.

De Maio et al. executed an analysis, PExy, on the source code of over 50 EKs in 37 families to recognize the conditions that make redirections to certain exploit and malware samples [11]. They also worked with EKs

in offline mode in their laboratories and via automated static source code analysis, where they produced all combinations of HTTP request parameters (in particular URL and User-Agent) that cause an EK to trigger an infection. Their goal was to achieve as many different types of exploits as possible and to reveal a potential zero-day exploit, if one existed. In this way, they retrieved 279 exploit samples including variants. They also understood the internals by showing that most of the EKs reuse source code from other EKs and even a new EK usually is based on another EK.

While accessing the source code of the current EKs is not realistic, getting the EK network traffic could be feasible. Therefore, detection of EK network traffic is vital today. The following studies involved machine learning or statistics to detect EK traffic behind the attacks and our study also focuses on EK families from this perspective.

Eshete and Venkatakrishnan [12] analyzed samples of 38 EKs with WebWinnow and identified content and structural features to model a set of classifiers. They locally installed EKs in a controlled setting and partly supported the dataset with 11 live EKs that were reported by the URLQuery[2] service. They labeled all URLs as EK rather than EK families. Their model was built with 500 benign and 500 EK URLs to detect EK traffics. They trained the binary classifier with 1117 benign and 512 EK URLs. The final objective of WebWinnow converges with PExy, which is to reinforce existing detection systems.

Taylor et al. developed a method to categorize EK flavors by detecting structural patterns in HTTP traffic [13]. Initially, they represented interactions between the victim browser and EK servers as known EK trees. In the detection process, their model builds a candidate tree from the request-response pairs of new infections and finds similar EK products with the weighted Jaccard index. During the analysis period, they built their own dataset by capturing 3800 h of real-world traffic via a honeyclient, which includes 28 EK instances. The comparison with the state-of-the-art techniques showed that while the system gets similar true positive rates, it reduces false positive rates by four orders of magnitude. The details of the patented application were discussed in Taylor's dissertation [14].

Stock et al. offered a prevention mechanism, Kizzle, in contrast to previous studies, which was specifically designed to identify four major EKs (Angler, Rig, Nuclear, SweetOrange) as they evolve over time and produce signatures that can be applied to antivirus engines or plug-ins of a Web browser [15]. The main objective is to autogenerate host-based structural signatures by the DBSCAN machine learning algorithm within hours for detecting superficial but frequent changes. They also observed that all JavaScript code served by EKs applies obfuscation and EK families reuse exploits from each other. While the packed view of the JavaScript code is unique across incidents, unpacked code is quite common (e.g., actual fingerprinting and CVE code). They generated the dataset in a 4-week period in August 2014. The evaluation showed that the false negative rates are under 5%, while false positive rates are under 0.03%.

There are also some studies where authors perceived the EK phenomenon differently from particular angles. Jayasinghe et al. [16] detected drive-by download attacks at runtime using lightweight dynamic analysis of the byte code stream generated by a Web browser during page content execution. They collected their dataset from forums that publish new URLs, which deliver malware. The approach extracted Opcode call sequences as features from the JavaScript engine of the Web browser, which generates Opcodes as a part of the rendering process for each webpage. They utilized naive Bayes, support vector machines (SVM), and decision trees as binary classifiers and SVM achieved the best score with almost 95% accuracy. Nappa et al. [17] identified drive-by download attacks by clustering exploit servers belonging to 2 different EKs based on 7 features related to the

[2] urlquery.net

served exploits and distributed malware. They utilized two clustering algorithms, which are partitioning around medoids and an aggressive clustering algorithm. According to the analysis, they observed a highly polymorphic ecosystem, where both exploit and malware files were packed differently in order not to be detected from the same file hash. They also made their generated dataset available to academic researchers. Sood et al. [18] conducted a comparative study for 10 EKs and found 3 victim profiling methods, which were based on user-agent fingerprinting, HTML document object model (DOM)-based fingerprinting, and IP-based geolocation tagging. There were 4 JavaScript-based attack techniques for drive-by download, which were obfuscation, redirection, content injection on the fly, and the domain address generation algorithm (DGA). Takata et al. [19] proposed a method, MineSpider, which analyzes JavaScript code relevant to browser fingerprinting and redirection functionality, then reveals URLs in the webpage by executing the extracted redirection code with the Rhino JavaScript interpreter. MineSpider was implemented in a browser emulator, HtmlUnit, that can emulate an Internet Explorer 6 browser on Windows XP SP2 and the Java Runtime Environment (JRE), Acrobat PDF, and Flash Player browser plugins for automatically extracting URLs from webpages independently from the analysis environment. Their malicious dataset contained over 19,000 URL addresses and was captured during a 3-year period with the high-interaction honeyclient Marionette. MineSpider extracted 30,000 URLs in a few seconds by applying program slicing to JavaScript code inside the malicious webpages that were previously detected as drive-by download attacks from 9 EK families. Aldwairi et al. [20] tested 23 machine learning classifiers using a dataset of 5435 webpages containing drive-by download attacks and based on the detection accuracy they selected the top five to build the detection model. They extracted 26 content features without executing the webpage and reduced the feature vector size to 15. The bagged trees binary classifier achieved the highest accuracy with 90%. The disadvantage of the study is that although malicious content is triggered via JavaScript, they do not render page content. The method provides execution time gains; however, essential dynamic features are not considered. Jagannatha [21] proposed a two-layer detection scheme for EKs and processed a Bro-IDS HTTP log of 1000 samples generated by a third party in 2012. Naive Bayes was applied for binary classification and then K-means was utilized for clustering EK families. The 36 features were reduced to 6 attributes and achieved 99% supervised and 75% unsupervised accuracy for 400 reserved samples. While this research does not work with network traffic, it relies on quite basic features, does not benefit from structural patterns in URLs, and ignores content features. Sandnes [22] extracted the URL addresses from the output of an IDS for EK activity detection. The system can detect the sample as either benign or malicious rather than detecting the EK family. A custom dataset was built for experiments by relying on the domain addresses, which were previously associated with an EK activity and triggered IDS alerts related to EK signatures. The SVM, random forest, and naive Bayes classifiers were utilized with 9 features, where random forest achieved the best accuracy with 97%.

As the source codes of the recent EKs are not available to the public yet, in our previous study, with Know Your EK [23], the webpage contents of EK families were explored, and our current research focuses on URL components of EKs. The present work, using IsEK, is similar to the latest three studies [12, 13, 15], which tried to distinguish between EK types using HTTP traffic. The approach in Kizzle [15] is closer to ours, where the EK families appearing in the evaluation significantly overlap with our EK set. On the other hand, that feature set is only based on page content and the authors reported that their clustering approach inherently requires large amounts of data. Some aspects of WebWinnow [12], such as the use of URL features, are also similar to IsEK. Unfortunately, WebWinnow requires a sandbox environment to extract basic content features and it is not easy to build an identical one for fair comparison. In addition, the honeyclient technology usage

in WebWinnow breaks scalability. However, we base our methodology on lightweight analysis with machine learning and utilize simple mathematical calculations and avoid using regular expressions while extracting URL features. Moreover, our method relies on multifamily classification, which is more informative when compared to their favored binary classification. In a nutshell, the proposed technique performs faster and is scalable via customized machine learning algorithms and does not require massive data. The developed models are accurate, achieving over about 91% for 3 supervised algorithms (KNN, SVM, GBC), which is evidence that our approach is estimator-independent. It is important to note that only URLs are leveraged to achieve such a capability.

3. Methodology

Previously, our approach involved context-aware content analysis [23] of the EK-based cases stored in pcap files. There are two key discoveries about EK characteristics. First, each EK has a similar work flow for malware delivery as illustrated in Figure 1. More precisely, infections contain 5 elements: the campaign, gate, landing page, exploit, and malware. Second, each component in an infection chain follows particular templates. For instance, the length of URLs fall within specific boundaries, URLs contain a peculiar number of query keys, and their values have tailored formats.

The innovation in this study is leveraging the overall URL patterns embedded in HTTP interactions between EK servers and victim machines. Specifically, instead of analyzing each URL independently, the goal is to inspect all URLs, which are posted automatically after one click and without any user consent, together. The structures in the work flow allow to characterize EK flavors to a certain extent. After evaluating the statistical differences of the URLs of the entire infection chains, we were able to design distinguishing features for each EK family. Conclusively, the approach takes advantage of machine learning techniques for the discrimination of network traffics that belong to EK-based infections. In this sense, the proposed method differs from two similar studies: the system in [12] that combines both URL and content features with binary classification methods and the work of [15] that only analyzes Web contents individually.

3.1. Data sources

Access to real-world EK data is usually restricted to companies, government agencies, and research institutions that have had their systems intentionally exposed to these attacks, not made available publicly. To the best of our knowledge, Kafeine[3] and Bradly Duncan[4] are the top contributors of open-source EK research data. Kafeine is usually the first expert who realizes totally new types of campaigns and EK families. The major contribution of Bradly Duncan is the captured network traffic files, which are shared on his website. On the other hand, generating our own data corpus may seem to be another option. Although it is not impossible, the task is quite difficult with some drawbacks. The advantages of using a community-driven data corpus over generating our own are that it enables proof of the study quality, provides acceptability by a larger audience, opens doors for future researchers to compare their own results, and offers high quality in the data utilized. To this end, the primary data source of this study is the full packet captures shared by Bradly Duncan, which is an advantage of the introduced study, while other researchers depend on private datasets. The origin of the traffic are incidents that have resulted in malware infection after exploiting a client-side vulnerability through various EK products. The network captures are stored in the industry standard pcap file format and are available via the public website. It is crucial that all samples were generated during 2016; hence, this study totally represents one year, which is also another exclusive aspect when compared to other studies. The EKs exhibit a significant

[3] malware.dontneedcoffee.net
[4] malware-traffic-analysis.net

evolution in a longer period of time, which makes detection difficult. We also include a data corpus[5] shared from a website for testing purposes.

The network traffics were sniffed while intentionally visiting the compromised webpage that causes malware infection through an EK at the end. The communication between the victim system and EK infrastructure is provided via real operating system and real browser personalities, contrary to the mentioned related work that usually relies on honeyclients.

It is imperative that such a study conduct offline analysis, since campaigns and hosted pages by EKs quickly disappear. In addition, offline analysis provides two benefits, which are repeatable experiments and acknowledgment of a broad audience. On the contrary, online analysis is not as dependable, since EK behaviors usually depend on client profiles and the EKs do not give the same response for every request. While exploits and malware change according to the victim environment, EKs present benign behaviors for certain end-user platforms. Therefore, while a researcher gets an infection, some others could get normal Web browsing. In that case, the evaluation and comparisons would not be fair.

3.2. Processing pcap files

We utilized two widely common tools to process pcap files in order to cross-check the results of one with the other. First, the Tshark library that is the command line interface behind the well-known network packet capture and analysis tool Wireshark[6] was utilized. The second tool executed is Bro[7], which was developed and maintained by the International Computer Science Institute at the University of California at Berkeley and supported by the US National Science Foundation (NSF). The objective is extracting HTTP traffic (URL and related metadata) and HTTP files, and assigning general labels to each URL. Although we focus on just URLs, the page contents were also extracted in order to be sure there is really a malware infection after exploitation.

3.3. Label confirmation

First, although the dataset provider is definitely reliable, all pcap files were manually analyzed and labels were confirmed. The training dataset comprises all the incidents that happened throughout 2016 and the total number of pcap files is 189. There are 30 incidents containing malicious spam (malspam), which are outside the scope. The EKs that have a small number of incidents such as Sundown EK (5), Magnitude EK (3), and KaiXin EK (2) were removed. There is one pcap file that has an infection from both Angler and Rig, which was discarded. Finally, 4 pcap files were also removed, where they contained EK-data-dump, Dridex, ISC-diary, and a malicious Android application. In total, 45 pcap files were discarded and the remaining set contains 144 infections from Rig, RigV, Angler, and Neutrino EK families that correspond to 1456 URLs. The test dataset covers the incidents that also happened during 2016. The total number of pcap files here is 96. The infections belong to Rig, RigV, and Neutrino EK flavors that involve 818 URLs.

The pcap files that contain corrupted HTML, exploit, or malware files due to several reasons (e.g., network fragmentation) were not discarded, although we are not able to recover them with industry standard tools by default settings, since we wanted to validate that the incidents under investigation execute at least one exploit and malware. In addition, we consider only the URLs in the infection chain rather than page contents; thus, there is no problem with invalid files.

[5] broadanalysis.net
[6] www.wireshark.org
[7] www.bro.org

3.4. A preliminary manual analysis of EKs

A URL address is a string that is placed to access resources hosted on the Web. There are three logical parts in a URL, which are hostname, path, and query, as shown in Table 1.

According to our key observations through manual EK analysis, there are significant structural patterns across EK infections. First, an attack usually starts with a campaign page, where the URL address does not contain a path or query parts. Next, the landing page, exploit, and malware files are served from the same domain address and frequently the URLs are relatively long. Finally, after malware is executed on the victim system, a reverse connection is established for command and control (C&C) activity via a third domain address that contains just a path in the URL without a query field.

Table 1. Logical characterization of a URL.

URL Format	<domain name>.<top level domain>/<path>/<query>
Sample URL	abc.mydomain.com/path1/path2/page.html?param1=val1&param2=val2
Hostname	abc.mydomain.com
Path	/path1/path2/page.html
Query	?param1=val1&param2=val2

The dominant characteristics of Neutrino EK infections are that the lengths of the URLs are not very long and not very short, URLs usually do not have a query part, and the path segment includes many dash characters. The incidents also have two specific characteristics. First, some chains start with a URL without any path or query, then follow 4 URLs from the same domain address that have only the path field. Some other cases start with a URL ending with a JavaScript filename, then follow 4 URLs from the same domain address. After that, one URL with a key-value pair in the query region appears, and finally one IP address is accessed ending with a filename for the C&C process.

The dominant characteristics of Angler EK infections are that the lengths of the URLs are long, there are at least several URLs per chain, URLs usually have many key-value pairs in the query part. The incidents also have two particular characteristics. First, some chains start with a URL without any path or query, then follow 5 to 7 URLs from the same domain address with or without path field, and after that a command and control URL with a filename and key-value pair in the query segment appears. Second, while a set of the cases contain many URLs for command and control purposes, the other cases access IP addresses with a filename.

The dominant characteristics of Rig EK infections are that the lengths of the URLs are long, including many dashes or underscores. The chains start with a URL without any path or query, then sometimes follow one or two URLs from the same domain address, where the first one has no filename but a path part. The next URL has a filename with a path segment, followed by 3 or 4 URLs from the same domain address, where the first one has no filename but a query field, and the next 2 URLs have a filename with a query region. Finally, one IP address or domain is accessed, ending with a relatively short path for C&C efforts. Some versions of Rig EK infections have a slight difference. The lengths of the URLs are long. The chains start with a URL without any path or query, then follow 3 URLs from the same domain address, where the first one has no filename but one key-value pair in the query part including many dashes or underscores. The next two URLs have a filename with one key-value pair in the query field including many dashes or underscores. Finally, one IP or domain address is accessed, ending with a relatively short path for C&C services.

The dominant characteristics of RigV EK infections are that the lengths of the URLs are long. The

chains start with a URL without any path or query, following 3 or 4 URLs from the same domain address, where URLs have no filename but 6 key-value pairs in the query parts including many dashes or underscores. Finally, one IP address or domain is accessed, ending with a relatively short path for C&C functions.

Table 2. Sample infection from RigV.

Functionality	URL address
Campaign	joellipman.com/
Landing page	add.ibeattheclockatticktock.com/?aqs=yandex.74p77.406f4y2&oq=CelqA8fMlKbsDOVbj3BOJLQ1mz48OVAkWpP2uikLTzB_IhJeH9CW9UU4HupE&sourceid=yandex&es_sm=100&q=z3rQMvXcJwDQDoTGMvrESLtEMU_OHkKK2OH_783VCZ39JHT1vvHPRAP2tgW&ie=Windows-1251
Exploit	add.ibeattheclockatticktock.com/?ie=Windows-1251&q=z37QMvXcJwDQDoTDMvrESLtEMU_OH0KK2OH_783VCZz9JHT1vvHPRAPwtgWCel&es_sm=129&sourceid=chrome&aqs=chrome.125x57.406a8x0&oq=qA8fMlKbsDOVbj3BOJLQBmz48OVAkWpP2rikLTzB_IhJeH_CWMYgpD_6LWU7dt
Malware	add.ibeattheclockatticktock.com/?aqs=edge.122a103.406k4r4&sourceid=edge&es_sm=91&q=w3bQMvXcJxfQFYbGMvLDSKNbNkbWHViPxoyG9MildZ-qZGX_k7rDfF-qoV_cCgWRxfE&oq=qfLZQNQHo3kHVeQMwyocLVVtA9vqo3UTQmkKYg5CE-BzZZQhF-qKSELk93VzFkrFUcw&ie=UTF-8
C&C activity	ffoqr3ug7m726zou.ihuk7s.top/0123-4567-89AB-CDEF-0123?iframe

In addition to the gained insights from the anatomic appearance of EK infections, we also identified internal concrete structures in URLs. For the sake of clarity, we support our claim with an example contained in the dataset as shown in Table 2. For this EK family, RigV, the landing page, exploit, and malware URLs have a query part but do not have a path field. There are 6 key-value pairs in the query segment and their order changes across URLs. While the query keys are also almost the same among different incidents, the values of the keys are diverse, which are also almost unique among different incidents. More precisely, for this particular infection, there is a 5-character key 'es_sm' and its value is a 2 or 3-digit integer (e.g., for exploit URL '129'). There is a 9-character key 'source_id' and its value has a pattern that indicates the browser vendor (e.g., for malware URL 'edge'). There is a 2-character key ('ie') and its value (e.g., for landing page URL 'Windows-1251') has a pattern that indicates the character encoding. There is a 3-character key 'aqs' and its value (e.g., for exploit URL 'chrome.125x57.406a8x0') has a pattern that has the browser vendor, a dot, a two- or three-digit number, a lowercase character, a two or three-digit number, a dot, a two or three-digit number, a lowercase character, a digit, a lowercase character, and a digit. There is a 2-character key 'oq' and its value (e.g., for malware URL 'w3bQMvXcJxfQFYbGMvLDSKNbNkbWHViPxoyG9MildZ-qZGX_k7rDfF-qoV_cCgWRxfE') has a pattern that is a minimum of 60 and maximum of 67 characters in a mixed-case alphanumeric string containing at least one dash or underscore special character. There is a 1-character key 'q' and its value (e.g., for exploit URL 'z3rQMvXcJwDQDoTGMvrESLtEMU_OHkKK2OH_783VCZ39JHT1vvHPRAP2tgW') has a pattern that is a minimum of 59 and maximum 67 of characters in a mixed-case alphanumeric string containing at least one dash or underscore special character.

3.5. Feature design

The integral issue is designing the attributes for the machine learning algorithms and coding them into numerical values. The most obvious technique could be searching for the patterns mentioned above: for example, whether

the given URL has 6 key-value pairs in the query part or whether the given URL contains a 5-character key that has a two- or three-digit number. The aforementioned technique involves pattern searching that is usually conducted with regular expressions. Such an approach is applied to detect just the target object, no less or no more, to prevent an excessive search space. Therefore, we deduce that in order to be less affected by the high potential changes in URL patterns, we should follow an intelligent approach that employs statistics. Counting tokens, measuring lengths, and calculating minimum and maximum values appears to be the optimal solution. Such mathematical operations are many times more efficient than pattern searching in terms of the time consumed and speed of action.

For the dataset, with respect to quantifying the patterns in URLs, first we measure the path length, count the path tokens, and calculate the maximum, minimum, and average of those tokens. Basically, in this way, a 20-character path that has one token is discriminated from a 20-character path that has five tokens. Secondly, we apply the same logic to the query part, but the key-value pairs are computed separately. Likewise, in order to differentiate EKs more reliably, counting the particular special characters, dashes, and underscores is also taken into account to recognize the minor changes of EK families. The extracted features include the following: path length, query length, count of path tokens, path minimum length, path maximum length, path average length, path sum length, count of query key tokens, query key minimum length, query key maximum length, query key average length, query key sum length, count of query value tokens, query value minimum length, query value maximum length, query value average length, query value sum length, count of special characters, count of URLs, and count of unique domain addresses.

A custom Python-based script was developed to extract features, especially statistics from the full URL addresses. The feature design decision is based on the analysis drawn from the live EK families that are hosted on the World Wide Web. The attributes were derived from 144 incidents of 4 distinct, currently dominant EK flavors. After the labels of the dataset were manually verified, 20 features were extracted for each URL from all infection chains. The output of the script is the actual dataset that will be subjected to machine learning where classification algorithms are applied to enable processing for high speed and accuracy.

3.6. Preprocessing features

In order to build accurate machine learning models, the raw dataset was purified, as in the first try the algorithms could not perform well. It is considered that transforming actual values of features into an explicit representation could improve machine learning estimators. In this scope, four common scaling methods were evaluated, which are the maximum and minimum scaler, standard scaler, standard normalizer, and binarizer. Experiments showed that the standard scaler performs best on the training dataset.

3.7. Models and experiments

3.7.1. Environment and instruments

Using the features extracted on the sanitized dataset, the scikit-learn machine learning API [24] is adopted to build classification models. Several classifiers have been tested; however some algorithms (e.g., linear and logistic regression, stochastic gradient descent, decision trees, naive Bayes) are not well optimized. In this study, we keep our focus on EK detection rather than the individual successes of machine learning algorithms, as replacing machine learning algorithms is quite easier than designing a method for detection. Therefore, we have selected 3 algorithms known for their high performance in terms of accuracy and execution time at the preelimination stage, which are KNN, SVM, and GBC.

Table 3. Dataset partitions for cross-validation.

EK Label	# Infections	# URLs
Angler	31	267
Neutrino	33	216
Rig	52	350
RigV	28	188
Total	144	1021

Table 4. Dataset partitions for testing set.

EK Label	# Infections	# URLs
Neutrino	35	221
Rig	55	386
RigV	6	41
Total	96	648

3.7.2. Hyperparameter optimization

Principally, machine learning methods follow formulations. KNN, SVM, and GBC have variables called hyperparameters, which could be tuned for better performance. In order to reach the capability limits of the methods, the hyperparameters are optimized based on the training dataset. The same stratified 5-fold cross-validation process is applied for all three algorithms in the optimization process.

3.7.3. KNN

The hyperparameter of KNN is k, which is the number of neighbors. The range for k is chosen as the odd numbers between 1 and 15. For every value of the hyperparameter, 5-fold cross-validation is applied. The optimum value of the hyperparameter k is 5.

3.7.4. SVM

The hyperparameter set for SVM is cost and class weight while the SVM kernel is linear. The hyperparameter set for the SVM is cost, class weight, and gamma while the SVM kernel is rbf. For every value of the hyperparameters, 5-fold cross-validation is applied by the grid optimization technique. The best hyperparameter set is that when the kernel is rbf, cost is 10, gamma is 0.001, and class weight is none.

3.7.5. GBC

The hyperparameter set for GBC is learning rate, number of estimators, and subsample. For every value of the hyperparameters, 5-fold cross-validation is applied by the random search optimization technique. The best hyperparameter set is learning rate of 0.8, number of estimators of 400, and subsample of 1.

3.7.6. Training

The goal of the training step is to evaluate designed features that are derived from the URL characterization of EKs. Using tuned hyperparameters for 3 supervised learning methods, customized KNN, SVM, and GBC models are built and the labeled dataset is used to train the classification models. Fivefold cross-validation, shown in Table 3, is utilized for each algorithm to measure the performance.

3.7.7. Testing

The aim of the testing phase is to measure the accuracy of the classifiers, while classification models group unknown infection chains according to their EK family. Table 4 summarizes the breakdown of infections in the test set.

4. Evaluation

This section discusses the evaluation of an efficient classification method by the application of machine learning techniques for state-of-the-art EK traffic detection. The accuracy of the estimators is assessed, the significance of the derived features is questioned via the cross-validation results, and the misclassified samples are properly justified. A comparison of the studies that apply similar techniques is also extensively presented.

4.1. Performance results

Our approach leverages the patterns of URLs appearing in infections based on EKs and the core of the proposed technique is the analysis of the URLs belonging to an incident altogether. The classification models were developed using 3 supervised learning algorithms (KNN, SVM, and GBC) and evaluated to decide which estimator is more suitable for EK discrimination. The first metric is the accuracy on the training dataset using 5-fold cross-validation and the performance of these classifiers for the training phase is illustrated in Figure 2.

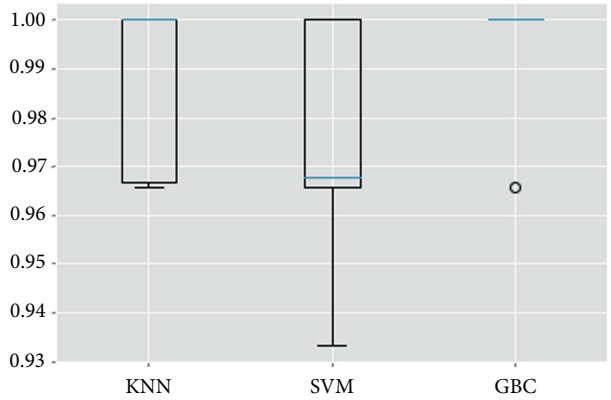

Figure 2. The performance of classification models with cross-validation.

The second metric is the accuracy of the designed models on the test set, which was obtained from a completely different source that enables to verify the quality of the models effectively. In the testing phase, the trained classifiers were independently executed and KNN, SVM, and GBC achieved 90.6%, 88.5%, and 98.9% classification accuracy, respectively. When we optimize our dataset by discarding excess C&C communication traffics, the models performed better and KNN, SVM, and GBC achieved 95.8%, 91.6%, and 100.0% classification accuracy, respectively. It is sensible to get one hundred percent accuracy, since we manually checked nearly 2000 URLs and also discarded unrelated file types (e.g., txt, images, some JavaScript).

4.2. Analysis of features

Although KNN and SVM does not expose the importance order of the features, GBC provides such information where the model gains more power and relies on it while predicting. The rank of the feature gains is as follows: count of query key tokens, query value maximum length, count of query value tokens, query length, path sum length, query value average length, path average length, count of special characters, query key maximum length, path length, path maximum length, query value sum length, path minimum length, and query key sum length. When the models are tested with the top 5 features among the ranked 14 features, promising results are measured; however, even a small accuracy decrease is not tolerated. On the other hand, the remaining 6

features, count of path tokens, query key minimum length, query key average length, query value minimum length, count of URLs, and count of unique domain addresses, were not leveraged by GBC. However, we observed a performance decrease for KNN and SVM when these 6 features were removed where we implicitly deduced that they were utilized somehow. Ultimately, all features were kept.

4.3. Error analysis

The model based on GBC detects previously not seen EK infections better than the other two algorithms. Only 1 sample was misclassified by the model. An infection from Rig was predicted as Angler by the classifier. SVM misclassified 11 samples and 9 of them were also misclassified by KNN. However, it is easy to justify these decisions. This is because there are uncommon command and control activities in these infections that cause many paths and tokens. Removing duplicate URLs that are usually seen in command and control activity could be a solution here, as well as discarding the URLs that exceed a limited number of URLs per chain.

4.4. Comparison

Although EKs have been researched in the past years, studies dedicated to EK detection are quite limited. Moreover, while our study utilizes machine learning for detection, other works mainly apply custom techniques. The results of the current analysis and the literature are compared in Table 5 to give an overall idea. Webwinnow [12] evaluated 5 binary classifiers and J48 performed better. When compared to our lightweight study, that method is quite time-consuming due to the examination of page contents rather than solely utilizing URL addresses. Taylor et al. [13] employed the weighted Jaccard index and also inspected both URLs and page content. While Kizzle [15] utilized DBSCAN for clustering web content, particularly JavaScript code blocks, the number of features is not a valid criterion for their model and only false negative rate (FNR) was reported. Jagannatha [21] only tried naive Bayes in combination with K-means and IsEK performs better in terms of accuracy. Sandnes [22] experimented with 3 classifiers and random forest achieved the best score, and the model was only able to detect samples being malicious or not. On the other hand, our proposed method discriminates particular EK families with a high accuracy.

Table 5. Comparison with the other studies.

Study	Accuracy	Features	Algorithms
[12]	TPR: 99.9% FPR: 0.001% FNR: N/A	30 Page content	J48 decision tree
[13]	TPR: 95% FPR: N/A FNR: N/A	8 Page content and URL	Weighted Jaccard index
[15]	TPR: N/A FPR: 0.03 FNR: %5	Page content	DBSCAN
[21]	TPR: 75%-85% FPR: N/A FNR: N/A	6 URL	Naive Bayes and K-means
[22]	TPR: 97% FPR: N/A FNR: N/A	9 URL	Random forest
Ours	TPR: 98.9% FPR: N/A FNR: N/A	20 URL	Gradient boosting classifier

5. Conclusions and final remarks

This research proposes a lightweight discrimination system for the network traffic of EK families. By using only the URL characteristics of a complete infection chain, our overall URL patterns technique reasons about the likelihood of a sequence of HTTP interactions belonging to a specific EK. Our implementation is evaluated on a real-world dataset collected by a pioneer researcher on EK. In particular, our empirical results show that

supervised model IsEK classifies EK families quickly with between 91.6% and 100% significant accuracy and a very low misclassification rate. The results validate our hypothesis that EK infections largely tend to have hidden patterns in URLs, which are only discovered via the analysis of overall URLs that are responsible for a successful malware infection. An individual URL analysis could not reason about whether a set of HTTP traces belongs to an EK infection or not, since every URL does not reflect an EK pattern. For example, some URLs do not contain either the path or query, i.e. they are just domain addresses and previously never seen in malicious activity, which also makes them blacklist-free. On the other hand, the proposed novel overall URL patterns technique is highly efficient in discriminating EK families.

It is assumed that such an agile solution will help security analysts who work with bulk data collected by honeypots through early discovering of the modified attack techniques to reveal zero-day attacks and also create obstacles and raise bars for cybercriminals and cause increases in the workload of EK engineers.

During the experiments, we evaluated many URL features but selected the attributes that are easiest to extract in terms of processing time. Some of the notable properties that are discarded for the mentioned reason included total number of HTTP GET and POST requests, total number of redirections, total number of distinct domain addresses, total number of unique country codes in domain addresses, total number of unique top level domains (TLDs), total number of distinct files downloaded to the victim system involved in the infection chain, count of some notorious mime-types (e.g., Shockwave file, Octet-stream, plain text), and total bytes of downloaded content to the victim system. While IsEK is based on the extraction complexity, the decision criteria could rely on purely a feature selection algorithm (e.g., information gain) in order to get better accuracy while reducing the number of features.

References

[1] Provos N, Mavrommatis P, Rajab MA, Monrose F. All your iframes point to us. In: Proceedings of the 17th Usenix Conference on Security Symposium; San Jose, CA, USA; 2008. pp. 1–16.

[2] Wang YM, Beck D, Jiang X, Roussev R. Automated web patrol with strider honeymonkeys: finding web sites that exploit browser vulnerabilities. In: Proceedings of the 13th Annual Network and Distributed System Security Symposium; San Diego, CA, USA; 2006. pp. 35–49.

[3] Provos N, McNamee D, Mavrommatis P, Wang K, Modadugu N. The ghost in the browser analysis of web-based malware. In: Proceedings of the 1st Usenix Workshop on Hot Topics in Understanding Botnets; Cambridge, MA, USA; 2007. p. 4.

[4] Seifert C, Welch I, Komisarczuk P. HoneyC - The low-interaction client honeypot. In: Proceedings of the New Zealand Computer Science Research Student Conference; Hamilton, New Zealand; 2007. pp. 1–9.

[5] Moshchuk A, Bragin T, Deville D, Gribble SD, Levy HM. SpyProxy: Execution-based detection of malicious web content. In: Proceedings of the 16th Usenix Conference on Security Symposium; Boston, MA, USA; 2007. pp. 1-16.

[6] Nazario J. A virtual client honeypot. In: Proceedings of the 2nd Usenix Workshop on Large-Scale Exploits and Emergent Threats; Boston, MA, USA; 2009. pp. 911-919.

[7] Zhang J, Seifert C, Lee W, Stokes JW. ARROW: Generating signatures to detect drive-by downloads. In: Proceedings. of the 20th International Conference on World Wide Web; Hyderabad, India; 2011. pp. 187–196.

[8] Grier C, Pitsillidis A, Provos N, Rafique MZ, Rajab MA et al. Manufacturing compromise: The emergence of exploit-as-a-service. In: Proceedings of the 19th ACM Conference on Computer and Communications Security; Raleigh, NC, USA; 2012. pp. 821–832.

[9] Kotov V, Massacci F. Anatomy of exploit kits: preliminary analysis of exploit kits as software artefacts. In: Proceedings of the 5th International Symposium on Engineering Secure Software and Systems; Paris, France; 2013. pp. 181–196.

[10] Allodi L, Kotov V, Massacci F. MalwareLab: Experimentation with cybercrime attack tools. In: Proceedings of the 6th Usenix Workshop on Cyber Security Experimentation and Test; Washington, DC, USA; 2013. pp. 1–8.

[11] De Maio G, Kapravelos A, Shoshitaishvili Y, Kruegel C, Vigna G. PExy: The other side of exploit kits. In: Proceedings of the 11th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment; Egham, UK; 2014. pp. 132–151.

[12] Eshete B, Venkatakrishnan VN. WebWinnow: Leveraging exploit kit workflows to detect malicious urls. In: Proceedings of the 4th ACM Conference on Data and Application Security and Privacy; San Antonio, TX, USA; 2014. pp. 305–312.

[13] Taylor T, Hu X, Wang T, Jang J, Stoecklin MP et al. Detecting malicious exploit kits using tree-based similarity searches. In: Proceedings of the 6th ACM Conference on Data and Application Security and Privacy; San Antonio, TX, USA; 2016. pp. 255–266.

[14] Taylor T. Using context to improve network-based exploit kit detection. PhD, University of North Carolina, Chapel Hill, NC, USA, 2016.

[15] Stock B, Livshits B, Zorn B. Kizzle: A signature compiler for detecting exploit kits. In: Proceedings of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks; Toulouse, France; 2013. pp. 455–466.

[16] Jayasinghe GK, Culpepper JS, Bertok P. Efficient and effective realtime prediction of drive-by download attacks. Journal of Network Computer Application 2014; 38: 135-49.

[17] Nappa A, Rafique MZ, Caballero J. The MALICIA dataset: Identification and analysis of drive-by download operations. International Journal of Information Security 2015; 14 (1): 15-33.

[18] Sood AK, Zeadally S. Drive-by download attacks: a comparative study. IT Professional 2016; 18 (5): 18-25.

[19] Takata Y, Akiyama M, Yagi T, Hariu T, Goto S. MineSpider: Extracting hidden URLs behind evasive drive-by download attacks. IEICE Transactions on Information and Systems 2016; 99 (4): 860-872.

[20] Aldwairi M, Hasan M, Balbahaith Z. Detection of drive-by download attacks using machine learning approach. International Journal of Information Security and Privacy 2017; 11 (4): 16-28.

[21] Jagannatha P. Detecting exploit kits using machine learning. MSc, University of Twente, Twente, the Netherlands, 2016.

[22] Sandnes J. Applying machine learning for detecting exploit kit traffic. MSc, University of Oslo, Oslo, Norway, 2017.

[23] Suren E, Angin P. Know your EK: A content and workflow analysis approach for exploit kits. Journal of Internet Services and Information Security 2019; 9 (1): 24-47.

[24] Pedregosa F, Varoquaux G. Scikit-learn: Machine Learning in Python. 2011.