FLASH: Is the 20th Century Hero Really Gone? Large-Scale Evaluation on Flash Usage & Its Security and Privacy Implications

Damjan Buhov[1], Julian Rauchberger[1], and Sebastian Schrittwieser[2]*
[1]*Institute of IT Security Research*
[2]*JRZ TARGET*
St. Pölten University of Applied Sciences, Austria
{damjan.buhov, julian.rauchberger, sebastian.schrittwieser}@fhstp.ac.at

Abstract

Although the Adobe Flash browser plugin steadily lost popularity throughout the last few years, Flash content still regularly appears when browsing the web. Known for its infamous security track record, Flash remains a challenge in making web browsing more secure. In this paper, we present a large-scale measurement of the current uses of Flash, based on a crawl of the top 1 million websites. The different types of measurements result in most detailed classification of Flash uses to date. In particular, special attention is payed to Flash usage related to user tracking, as well as to malicious Flash files used by malvertising or exploit kits. We present Garrick, a novel crawling framework, which is based on a full-fledged Mozilla Firefox browser. Garrick is able to mimic any browser, plugin and operating system configuration so that fingerprinting scripts can be tricked to deliver malicious Flash files. Our measurements show that Flash is still used by approximately 7.5% of the top 1 million websites, with 62% of the Flash content coming from third-parties such as ad networks. In general, on popular websites Flash usage is higher compared to less prominent websites and a bigger share of Flash content on these sites comes from third-parties. From a security perspective, malicious Flash files served by highly targeted malvertising campaigns are an ongoing challenge.

Keywords: Adobe Flash, Malvertising, Exploit-Kits, User Tracking

1 Introduction

Integrating dynamic and multimedia content such as videos to a webpage basically required the use of the Adobe Flash platform for many years. This began to slowly change when the HTML5 standard introduced modern alternatives to the plugin-based Flash and operating systems as well as browsers limited their support for Flash.

The popularity of Flash also attracted criminals who use vulnerabilities in the Flash browser plugin to attack end-users. In the past few year, severe vulnerabilities were discovered in the Adobe Flash platform on a regular basis. For example, 266 new CVEs have been assigned to the Flash Player in 2016 alone[1]. Quite often, criminals use these publicly disclosed vulnerabilities in their exploit kits. Based on an analysis conducted in 2015, Adobe Flash Player comprised eight out of the top 10 vulnerabilities used by exploit kits [1]. In 2016, the Flash Player still comprised six of the top 10 vulnerabilities used by exploit kits [2]. Furthermore, according to the newest Adobe Security Bulletin, Flash player still contains critical vulnerabilities which need to be addressed immediately [3, 4]

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), 9:4 (Dec. 2018), pp. 26-40
*Corresponding author: Department Informatik & Security, Fachhochschule St. Pölten GmbH, Matthias Corvinus - Straße 15, 3100 St. Pölten, Tel: +43/676/847 228 648
[1]`http://www.cvedetails.com/product/6761/Adobe-Flash-Player.html?vendor_id=53`

In this paper, we present a report of a large-scale measurement of the current uses of Flash, based on a crawl of the top 1 million websites. The different types of measurements include the overall Flash usage by page popularity, first-party vs. third-party Flash content and the most common categories of websites using Flash. In particular, we set a focus on Flash files related to user tracking, as well as on malicious Flash files from malvertising campaigns or exploit kits embedded into website. We further analysed trackers and malicious Flash files that were collected while crawling the websites.

To perform the measurement, Garrick, a novel crawling framework was implemented. Garrick automates a full-fledged Mozilla Firefox browser, together with an orchestration framework that is used to deploy and run the infrastructure. The framework supports massive parallelism for speed and scale, and reports the results back to a central server. One of the main efforts was to ensure that the framework is able to mimic any browser, plugin and operating system configuration so that fingerprinting scripts can be tricked to deliver malicious Flash files.

In summary, we make the following contributions:

- **Garrick.** We present a novel crawling framework which is able to mimic any browser, plugin and operating system configuration.
- **Large-scale measurement of the current uses of Flash.** Our measurements of the top 1 million webpages result in the most detailed classification of Flash use to date.
- **Analysis of Flash based attacks and trackers** We analyse what Flash based trackers and attacks are most prevalent in the top 1 million webpages and discuss the challenge of identifying highly targeted attack campaigns.

The rest of the paper is structured as follows: Section 2 provides an overview of the Flash ecosystem and its attack vectors. The crawling framework Garrick and the sample sets for its evaluation are described in Section 3. Section 4 presents the measurement results with the different classifications of Flash uses, including Flash related trackers and malicious files. Section 5 discusses related work and Section 6 concludes the paper.

2 Adobe Flash Ecosystem

2.1 A short history of Adobe Flash

Flash was created by Macromedia in 1996 to ease the creation of animation on the web. Macromedia distributed the player for flash content as a free browser plugin to quickly gain market share. In 2000, Flash 5 was released and it added support for a new technology called ActionScript, an ECMAScript-compliant language, which makes ActionScript closely related to JavaScript. By adding advanced scripting capabilities, Flash became a flexible environment to run external code on client computers and create dynamic content. In 2005, Adobe purchased Macromedia and rebranded it to Adobe Flash. As Flash matured, it shifted from an animation and authoring tool to an integral part for enabling multimedia across the web. The popularity of Flash increased rapidly until it was ubiquitous on desktop computers with a market penetration of approximately 99%.

With the introducement of the HTML5 standard, Flash began losing its popularity mainly because of the fact that HTML5 can provide the same functionalities as the Adobe Flash with one big advantage, namely, eliminating the need of additional plugins.

In July 2017, Adobe announced the end-of-life of Flash in 2020 [5]. The end-of-life also means that no security updates for the Flash Player will be published after 2020. However, discontinuing the support for Flash doesn't remove Flash entirely. By 2020, legacy systems that still run Flash will be at high risk as newly discovered and unpatched vulnerabilities may get exploited.

Nowadays, Flash is still an important factor to consider when it comes to web security. Adobe states that the Flash Player is still used on more than a billion desktop computers [6]. An evaluation of the BuiltWith top 1 million webpages shows that approximately 2.4% are using flash content [7]. A different measurement by W3Techs concludes however that Flash is used by 4% of Alexa's top 1 million webpages [8]. Furthermore, they observed that the usage of Flash on all webpages is declining by 1% every 6 months.

2.2 Main uses of Flash

When Flash was widely installed on desktop computers, it was commonly used in a wide area of applications, such as to display interactive web pages, online games, and to playback video and audio content. However, the usage of Flash strongly decreased in most of these areas as it gets slowly replaced by other technologies.

Video streaming platforms mostly replaced their Flash-based players in the previous years. For example, Youtube replaced Flash in favor of HTML5 for its default web player as one of the first video platforms in 2015 [9]. But at the time of writing, there are still some streaming platforms which require Adobe Flash, e.g. HBO. One reason why not all streaming platforms dropped Flash may be the fact that Flash offers a content protection deeply integrated into the delivery platform and the file format.

Not long ago Flash-based ads were ubiquitous across the web. Nearly 84% of banner ads used Flash in 2015 [10]. Google stopped serving display ads in the Adobe Flash format on the Google Display Network and through DoubleClick in January 2017. Furthermore, the Interactive Advertising Bureau (IAB) set the goal to completely eliminate Flash Video Ads by July 2017 [11]. Great efforts are made to stop Flash-based ads, however, as our measurements will show there are still advertising networks left that serve them to the end-users.

Adobe Flash has also been heavily used in casual or browser games. Flash-based games are still used on the social media platform Facebook. Facebook announced that Flash-based games will run on their website until the end of 2020 [12].

With the programming capabilities of Flash, which were introduced via Action Script, not only browser games, but also Web-based applications were developed. These web applications are also known as "Rich Internet Applications" (RIAs). The trend of the previous years shows that also Web-based applications are shifting away from Flash. However, some management applications like Cisco Configuration Professional or the Web Client of VMware vSphere still require Flash Player to run. If all applications are not migrated to a different technology until 2020, we will face a vast number of unpatched legacy systems which run the Flash based applications. These legacy systems will then became an attractive target for the criminals due to the large set of unpatched vulnerabilities they will contain.

2.3 Flash-based Attacks

Adobe Flash has been abused by malicious actors to facilitate attacks on users for a long time. Due to its high market share, Flash has historically been an extremely worthwhile target. Even though it is no longer as widespread as it used to be a few years ago and modern security techniques such as sandboxing have increased the complexity of the attacks, Flash is still one of the main vectors targeted by exploit kits and malvertising campaigns.

At the time of this writing, cvedetails.com lists over 800 vulnerabilities for Adobe Flash Player in the Code Execution category alone. An environment where attacks can be implemented in a scripting language allows reliable exploitation of even minor memory corruptions as techniques like Heap Grooming and dynamic calculation of randomized addresses can often be implemented a lot easier than in data-only

attack scenarios. Both the relative ease of finding new vulnerabilities as well as the high reliability of attacks have ensured the relevance of Flash Player exploits over the years.

Even though, ActionScript has also been used for less complex attacks like injecting JavaScript in the page it is hosted on, or redirecting users to phishing websites, the most devastating vulnerabilities are mostly memory corruptions. These allow malicious actors to execute arbitrary code on the victim's machine and are often the first step in a chain of attacks that recruit the unsuspecting user's PC into a botnet or install ransomware, backdoors or other malicious software.

From a technical perspective, the core principle of exploiting Flash Player memory corruptions has not changed much. First, the attacker has to find a bug in one of the many parts of Adobe Flash that allows them to change the contents of memory outside the scope of their script. For instance, an ActionScript program is not supposed to be able to change the metadata associated with an array. If attackers can find a way to do it, they could overwrite the length field to bypass checks that normally stop the program from accessing memory outside the memory range which has been initially allocated for the given array.

Once attackers find a reliable way to manipulate the memory in a way that has not been intended by the developers, they can use this to manipulate the flow of execution. One common technique is to overwrite a function pointer within the table that holds the addresses of all functions that can be called on an object. If an attacker can change one of these pointers and subsequently invoke the function it corresponds to, he can direct the flow of execution to an arbitrary address.

Once the control of the instruction pointer has been achieved, techniques such as return oriented programming [13] or JIT spraying [14] can be employed to execute arbitrary shellcode.

Now, the attacker has full control over the process in which the ActionScript virtual machine had been hosted. Historically, this was mostly equivalent to full access to the machine or at least the user's account. Nowadays many browsers feature additional sandboxes which limit the impact of these attacks. Employing the principle of least privilege, the attacker-controlled process is still greatly limited in which other parts of the system it can access. Of course, sandboxes are only additional hurdles for attacker and many ways to break out of them have been and continue to be found.

Implementing a reliable exploit that works across many different platforms and configurations is still a challenging task. To extend the lifespan of their carefully crafted exploits, malicious actors employ a wide range of different techniques to avoid detection by antivirus software and manual reverse engineering.

A very widespread one is fingerprinting. Before attempting the actual exploitation of a memory corruption vulnerability, a harmless SWF that performs various checks is executed. The complexity of these first stage payloads varies greatly from simple operating system and flash version number checks to more complex techniques like font enumeration. Only if a potential victim is found to be vulnerable is the actual attack executed. This second stage is often completely separated from the first to ensure that exposure of the real malicious code to potential analysts and antimalware software is minimized as much as possible.

Analysis of exploits is often further complicated by obfuscation. Class, function and variable names are changed to unprintable garbage and nonsensical jumps and loops are introduced to confuse reverse engineers. Also very widespread is the use of packers which embed the actual payload as binary data inside the SWF and then decrypt it with a custom obfuscated routine at runtime before loading it dynamically.

It is obvious that malicious actors are still investing quite a lot of resources to implement Flash Player based exploits and hide them from security researchers and malware analysts. Even though key-players such as Google and even Adobe themselves have attempted to deprecate and remove Flash Players for quite some time now, the attack landscape shows that criminals still deem Flash widespread enough to be worth targeting.

2.4 User-tracking

Throughout the past years, tracking of users found its place in the industry as a successful revenue generator and established itself as a quite ”stable” business model. Apart from the numerous tracking methods used nowadays, the use of flash-based controls could be rendered as one of the most wide spread tracking method, due to its apparent presence manifested via the different use-cases mentioned in Section 2.2.

Similarly to HTTP, the most common representative of the flash-based controls is the *Local shared object - LSO* or popularly referred to as *cookie* stored on the users' machine. Even though, cookies can increase the usability of the web sites, they are often criticized by the security community due to the security risks they present. Unlike the traditional browser cookies, Flash cookies are relatively unknown to the web users and more importantly, they are not controlled by the browser cookie controls. This implies that even if a user has already cleared their browser of tracking objects, the reality is that they most likely have not done it. This fact, and the fact that the users are becoming overwhelmed with different advertisements carrying additional controls to monitor the activity, enables us the possibility to question the actual security and privacy in the web sphere and more importantly, it reveals the real size of user-profiling [15, 16, 17].

To counter this situation, a lot of companies are offering particular third-party applications and add-ons which claim that it solves the problem. In order to verify this fact, Merzdovnik et al. [18] performed a large-scale study of the most popular tracker-blocking tools to date. Their findings clearly outline that only small number of extensions are actually effectively blocking the majority of the trackers. Furthermore, they address another very important topic, namely, the lack of effective privacy preserving methods for mobile devices where user-tracking is far easier. Another large-scale study performed by Eckersley [19] revealed that 94.2% of the browsers which have Flash and Java installed can be accurately identified.

3 Framework and Sample Sets

3.1 Crawling Framework

Garrick consists of two major parts that together create our crawling framework. An automated browser scans websites and reports the results back to a central server and an orchestration framework is used to deploy and run all required infrastructure.

Figure 1 shows an overview of the components working together to ensure high performance in our crawling framework.

The web browser is based on SlimerJS[2], a JavaScript based framework to automate Mozilla Firefox. SlimerJS was chosen in favour of PhantomJS because of its ability to run browser plugins. One of the main challenges was to ensure that our Firefox browser behaves similar enough to the browser, plugin and operating system configuration desired, so that fingerprinting scripts cannot detect the difference.

By making use of XPCOM objects in the Firefox Add-on API, code can be injected into the JavaScript sandbox at the time a new document object is created. This code then patches various properties such as user agent, screen size, platform and language. Additionally, the `toString` representation of some objects is modified to match the targeted browser. Furthermore, the modified JavaScript environment reports an old, outdated version of Adobe Flash as well as ActiveX being installed if an Internet Explorer is mimicked. Finally, it hides some functions defined by SlimerJS that would give away the presence of an automation tool. The injected code is also used to collect data which is later sent back to the collection

[2]https://slimerjs.org

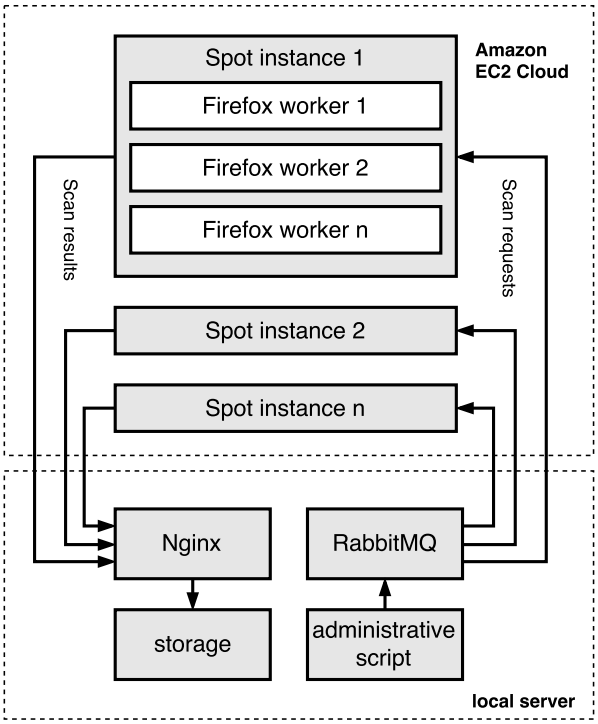


Figure 1: Components of Framework.

server. Built upon SlimerJS, our script can rely on existing APIs to gather information about various resources loaded during runtime. Data collection is stopped when a page has not loaded any new resources for a certain timeframe or after a hardcoded timeout is reached.

Some SWF files might load other files that are relevant to our research at runtime. For instance, multi-stage exploits where one SWF loads and executes another. To be able to detect and collect these resources, we decided to include a real Adobe Flash Player in our setup so Flash-based programs can be executed. We used a current version without any known security vulnerabilities but patched ActionScript to report an outdated version. This ensures that first-stage loaders that check the version of Flash Player before loading the actual exploit will be tricked into thinking that our browser is vulnerable and therefore load the second stage which we can then analyse.

To make ActionScript report an outdated version when accessing `Capabilities.version`, we simply replaced the version string directly in the Adobe Flash shared object. As long as the fake version number has the same length as the real one, we found this to cause no issues. Additionally, we wanted to make sure that accessing `Capabilities.os` would report the desired operating system, such as Windows 7 rather than Linux. We wrote an `LD_PRELOAD` based userland rootkit that intercepts calls to the `uname` function and patches the returned result to implement this feature. We found that ActionScript directly passes the data returned by `uname` to the calling ActionScript code, therefore reporting back the mimicked OS.

The browser is started by a Python script that receives the domain names that should be scanned from a central server using RabbitMQ[3], an open source message broker. It then loads the front page as well as two randomly picked subsites on the same domain and records, among others, data about all loaded resources as well as the full SWF files. When the scan is completed or after a certain timeout has been reached, all collected data is sent back to a central collection server over HTTPS. Resetting the browser

[3]http://www.rabbitmq.com

after each scanned domain ensures that data such as cookies is deleted and cannot influence subsequent scans.

One of the main goals was to make the framework scalable, so we decided to package the browser into a Docker container. This allows us to run multiple instances on the same machine without having to worry about them interfering with each other. Additionally, scaling the number of browsers up or down is very simple and additional Docker servers can be added quickly to increase capacity.

An administrative script adds domains that should be scanned to a RabbitMQ queue which is then distributed between all active Docker containers, which perform the actual scans. After each scan, results are sent back to a central server via HTTPS. Should one of the Docker containers not report back after a given amount of time, that scan is redistributed to another container.

To deploy this setup, we implemented an Ansible playbook that automatically installs all required services such as RabbitMQ, Docker and Nginx to the servers in our infrastructure. This allows us to easily recreate the full configuration and run production and development environments in parallel with minimal administrative overhead.

While we hosted the central management and collection server ourselves, we ran the Docker containers on EC2 Spot instances in the Amazon cloud. This allowed us to easily add resources for bigger scans while at the same time keeping expenses minimal during development. Even though Spot instances might be shut down at any time, this was no issue for our framework as work can easily redistributed to other containers in this case.

3.2 Sample Sets

We run measurements on the top 1 million web sites to provide a comprehensive view on the different types of flash usage. The sample set is taken from the Alexa top 1 million site list, which ranks sites based on their global popularity with Alexa Toolbar users. Prior running the complete sample set, small sub sets were used to test and improve the capabilities of Garrick.

When designing the framework, special attention was paid to be able to receive malicious Flash files. VirusTotal[4] is used as oracle to classify Flash files as malicious. The main issue of receiving malicious Flash files is that exploit kits serve Flash exploits only to systems that pass several fingerprinting checks.

Fingerprinting a system usually consists of two parts: server-side fingerprinting and client-side fingerprinting scripts. To address client-site fingerprinting scripts, the browser of Garrick was masked to look like an Internet Explorer 11 running on a Windows 7 system with a vulnerable Flash player plugin installed (see Section 3.1). To verify if the browser masking is successful, fingerprinting scripts of RIG, Terror and Magnitude exploit kits were analysed by visiting various infected pages. Also, it turned out that hooking JavaScript functions to record when fingerprinting scripts accesses specific information from the browser is not a useful oracle for potentially malicious sites. The same fingerprinting techniques are also used in other use cases, for example by various legitimate advertisements.

When it comes to server-side fingerprinting, the IP address is beside the user agent and the language settings the most important value that is looked for. Attackers do not want to expose their exploit code to others than potential victims. Therefore, they keep blacklists of IP addresses containing search engine crawlers, security companies, VPNs and cloud providers. Only potential victims with residential IP addresses not on their blacklist will get redirected to the next stage of an attack. Such blacklists are kept behind web services controlled by the attacker, so an infected site does not reveal its content.

The framework was tested by using different IP addresses on malicious sites to compare the addresses for their acceptance at server-side fingerprinting. Residential IP addresses showed the best result as the framework was not redirected to a clean site in the server-side fingerprinting stage. Running the

[4] `https://www.virustotal.com`

framework with IP addresses from VPN providers showed a totally different behaviour. If a VPN node does not have a residential IP address but is running at a big hosting provider, then only clean sites are served. The third option tested was using IP addresses from rented Virtual Private Servers (VPS). Here, IP addresses from smaller hosting providers in countries that are not known for running big datacenters, e.g. Spain, showed better results than the IP addresses from VPN providers or from big VPS providers in the US or Germany.

We have performed four distinct tests in order to be able to evaluate the scanning performance of the framework. Each test run scanned 10,000 random websites of the Alexa top 1 million site list and uploaded the resulting flash files to VirusTotal. Each scan was performed deploying 20 docker container on a EC2 Spot instance in the Amazon cloud. For comparison, each test run used two different IP addresses for scanning the websites, a small VPS provider's IP address versus Amazon's IP address. The scan results of the two different IP addresses show the same results, except for the second scan. In the second scan two malicious files were found with an Amazon IP address, compared to only one malicious file with the VPS provider's IP address. The results indicate that some blacklists for server-side fingerprinting do not cover all the IP addresses of Amazon's spot instances. Due to the similar results of the malicious files found and the bandwidth limitation of the VPS provider, the Amazon Spot instance's IP addresses were chosen for the large-scale scan. By using 50 m4-xlarge spot instances with each instance running 10 docker containers, the Alexa top 1 million websites can be scanned in around one and a half days.

It is noteworthy to state that as the time nears to the end-date for Adobe Flash, there is significant decrease in the numbers of malicious files that use the Flash plugin. This is consistent with the findings of different malware analysts that the activity of the most prevalent Exploit Kit at that time, started to significantly drop as of May 2017 [20, 21].

4 Results

The scope of our measurement study was the Alexa top 1 million web sites. The large scale allows us to answer different questions about the current prevalence of Flash, as well as it gives insights on the usage of Flash for malicious or fingerprinting purposes. Garrick collected 922.890 valid results, because not all web sites of the Alexa top 1 million list were available. The scan results contain 131.782 Flash files, of which 46.024 files are unique samples.

4.1 Uses of Flash

In our measurement 69.294 websites of the scanned sample set serve Flash content. This makes a overall Flash usage of approximately 7.5%. Figure 2 shows the Flash usage of websites dependent on the website popularity. We can see that popular websites use Flash content more often than less prominent sites. The Alexa Top 10k websites represent the sites with the most frequent Flash usage with approximately 10.6%.

Flash content can either be included directly on a website or being served by a third-party. From the collected Flash samples, 81.961 files, or approximately 62%, are coming from third-parties. Figure 3 shows the first-party Flash usage dependent on the website popularity. We can see that popular websites serve third-party Flash files more often than websites with a higher Alexa rank. First-party Flash files are least often used in Alexa Top 10k websites, with a share of approximately 28%. The reason why third-party Flash files are used more often on popular websites is because websites with a high number of visitors often use content delivery networks (either internal or external services).

Table 1 shows the 10 most common domains delivering the Flash content on websites of the Alexa Top 10k. By far the most Flash files came with 23.7% from *adap.tv*, a domain for online ads. The

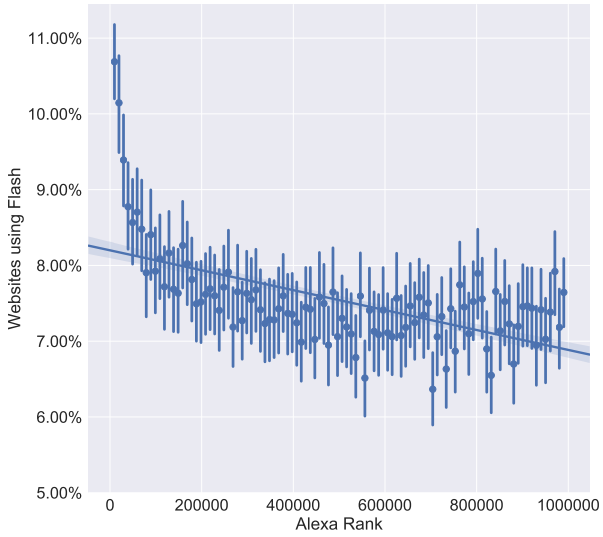


Figure 2: Flash usage dependent on Website popularity.

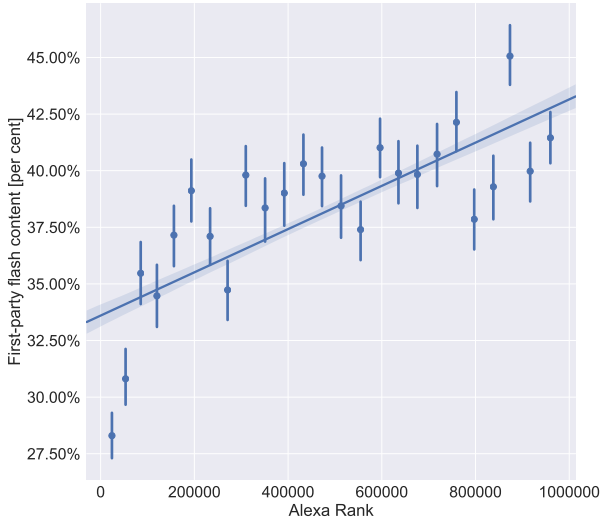


Figure 3: First party vs. third party content.

second most Flash files were served by *cmbestsrv.com* (12.1%), which is categorized as media streaming domain. The other domains in the top 10 are also a mix of online ads and media streaming domains. These domains served significantly less Flash files than the two most common domains.

The website categories of sites that use Flash are shown in Table 2. The left side of the table shows the categories for all websites serving Flash, while the right side of the table shows the categories only for Flash serving sites in the Alexa top 10k. The top category over all Flash serving sites is "Blogs/Wiki". The categories of popular websites is led by "General News", while "Blogs/Wiki" is only on rank 5. Figure 4 shows the ratio of first- to third-party content all websites serving Flash, grouped by website category. The top category "Blogs/Wiki" has the highest share of Flash content included directly on the websites.

Further, we used the Ghostery browser extension as classifier for Flash files coming from online advertisments or trackers. 14.4% of the Flash content was identified to come from advertisment or

	Domain	Percent	Sum
1	adap.tv	23.688	23.688
2	cmbestsrv.com	12.1356	35.8236
3	stickyadstv.com	5.77017	41.5938
4	zencdn.net	3.26774	44.8615
5	jwpcdn.com	3.21914	48.0807
6	growfoodsmart.com	2.74538	50.826
7	moatads.com	2.69679	53.5228
8	streamrail.com	2.33236	55.8552
9	googleapis.com	2.19874	58.0539
10	admicro.vn	2.00437	60.0583

Table 1: Domains delivering Flash content.

Category (All)	Percent	Category (Top 10k)	Percent
Blogs/Wiki	11.11	General News	23.53
Entertainment	8.92	Entertainment	11.85
Business	8.65	Games	8.44
General News	7.57	Portal Sites	5.27
Online Shopping	6.11	Blogs/Wiki	3.46
Education/Reference	5.77	Pornography	3.33
Pornography	5.52	Finance/Banking	3.25
Marketing/Merchandising	5.42	Internet Services	3.12
Games	4.76	Sports	3.04
Internet Services	2.93	Business	2.87

Table 2: Website categories of sites that use Flash.

trackers. Looking at the Alexa top 10k websites the amount of advertising and trackers raises to 20.9%.

4.2 Attacks

To classify Flash files as malicious, VirusTotal was used as oracle. 1655 files with at least one positive result were found. Of these files, 50 had at least 4 positive results and were further analysed manually. The manual analysis could confirm only 4 unique Flash files that show a malicious behaviour. The other files are most likely false positives, detected by some of the anti-virus engines because they either used packers or were otherwise obfuscated.

Three of the malicious files are so called droppers, which inject JavaScript to the website. The injected JavaScript performs fingerprinting checks, and if they are passed successfully, redirect to an exploit kit. All three malicious droppers have been known to VirusTotal since 2015 at least. Table 3 shows the websites, together with their Alexa rank, that spread the files. The malicious droppers were more frequently served on very uncommon sites with a Alexa rank higher than 700.000. The websites include the malicious droppers via `iframes`, except for *themoviedistrict.com*, which was hosting the file directly on the site.

The fourth malicious file files is an actual exploit of the Flash player, which is used by the Rig exploit kit. The exploit was served on the two websites *nontonfilm77.com* (Alexa rank 15.062) and *nontonmovie77.com* (Alexa rank 28.401). These sites have a much better Alexa rank than most of the Flash droppers found and are therefore more frequently visited. At the time of crawling the website, the

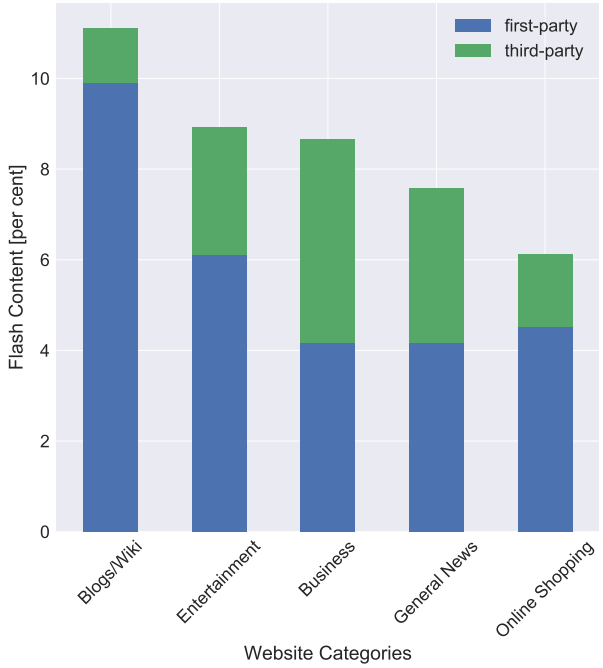


Figure 4: First party vs. third party content by Category.

Domain	Alexa rank
eeeqi.cn	31.351
mrmodern.com	149.582
emu618.net	723.125
rrkd.cn	834.675
138gzs.com	862.151
themoviedistrict.com	938.324

Table 3: Domains spreading Flash Droppers.

file containing the Flash exploit was uploaded to VirusTotal for the first time. The exploit is embedded on the site via an `iframe`. The `iframe` loads its content directly from the IP address *188.225.76.133*, without involving a DNS resolution.

Although we have put high effort into making the framework a valuable target for Flash exploits, only a small amount of malicious Flash files were collected while crawling the websites. This matches the trend that attackers are shifting away from using Exploit Kits on hacked websites to selectively use them in highly targeted malvertising campaigns. To catch Flash exploits from malvertising campaigns, a pool of residential IP addresses in different countries would be required, so that the framework matches exactly the target region of the malvertising campaign. Getting access to a large number of residential IP addresses, however, is highly challenging and was out of scope of our research.

4.3 Tracking

To identify Flash files that are used for tracking a user, we used two different oracles. The first oracle is the database of regular expressions of the Ghostery browser extension, and the second are the regular

expressions for Flash tracking files of the FPDetective framework [5], developed by Acar et al. [16]. With these oracles, 7.324 files, or approximately 5.5% of all collected Flash files, were found to be related to user tracking. The set of tracking Flash files contains 1.993 unique samples.

Tracker	percent	Domain	percent
FACETz	51.4462	facetz.net	51.3336
Wistia	20.2193	akamaihd.net	13.295
ReputationManager	9.11583	iesnare.com	9.09588
ThreatMetrix	4.89376	wistia.com	6.94843
iResearch	4.40027	online-metrix.net	4.88305
BlueCava	2.7279	irs01.net	4.39064
AOL CDN	1.80946	bluecava.com	2.72193
SITEBLACKBOX	1.80946	aolcdn.com	1.8055
Trustev	0.685401	trustev.com	0.683901
Sift Science	0.644277	siftscience.com	0.642867

Table 4: Tracker Categories and Domains.

Table 4 shows the most prevalent trackers using Flash, as well as the most common domains serving these tracking Flash files. The two most used trackers that include Flash files are *FACETz* and *Wistia*, with a share of over 70% of all collected trackers. *FACETz* is loading an evercookie (*https://front.facetz.net/evercookie.swf*) and *Wistia* is offering an embedded video player (*https://embedwistia-a.akamaihd.net/flash/embed_player_v2.0.swf*). The domains loading the trackers show that the evercookie is loaded directly from *facetz.net*, while the embedded player from Wistia is mostly loaded from the contend delivery system Akamai.

5 Related Work

Our survey on the uses of Adobe Flash covers different aspects: (a) An analysis how widespread the usage of Flash is across our sample of websites. (b) Which attacks (c) and which fingerprinting techniques that involve Flash were observed while scanning these websites. In this section we discuss recent work of all three fields in relation to our measurements.

Distribution of Flash. No scientific literature could be found that discusses how widespread the usage of Flash on current webpages still is. However, there exist Web services that measure the current usage of Flash, such as the services from BuiltWith [7] and W3Techs [8]. An evaluation of the BuiltWith top 1 million webpages shows that approximately 2.4% of the webpages are using flash content. The measurement by W3Techs concludes that Flash is still used by 4% of Alexa's top 1 million webpages.

Malicious Flash. Malicious Flash files that are used in attacks on end-users were analyzed in several studies. Ford et al. [22] focused in 2009 on the detection of malicious flash-based advertisements by using dynamic and static analysis techniques. The Alexa top 500 websites were crawled and approximately 0.15% of the flash advertisements were detected as malicious. Overveldt et al. [23] presented FlashDetect, an offline Flash file analyzer that uses both dynamic and static analysis, and that can detect malicious Flash files using ActionScript 3. More recently, Wressnegger and Rieck [24] inspected the properties and temporal distribution of Flash-based malware over a period of three consecutive years and 2.3 million unique Flash samples. They showed that Flash-based malware was constantly observed throughout the years with peaks that relate to popular attack campaigns. Attack campaigns that utilize

[5] https://github.com/fpdetective/fpdetective/blob/master/src/crawler/fp_regex.py

exploit kits with Flash exploits (amongst other exploits) are frequently used for distributing malicious advertisements over ad networks, as studies on Malvertising show [25, 26].

Flash Fingerprinting. Several studies examined the uses of Adobe Flash for Web tracking. Soltani et al. [27] first discovered the persistent tracking technique of flash cookies that re-instantiate HTTP cookies. Nikiforakis et al. [15] found that Flash is used for font extraction, to circumvent HTTP proxies set up by the user, and to get more fine-grained information such as the specific operating system kernel version or the presence of multiple-monitor setups to fingerprint a device. Acar et al. [16] implemented the FPDetective crawling framework and observed that 145 sites of the top 10,000 sites deployed Flash-based fingerprinting. In recent years, research in various methods of Web tracking was made with a decreasing importance of the Flash technology [28, 17].

6 Conclusion

Although Adobe is planning to remove support for Flash by the end of 2020, Flash content still regularly appears when browsing the web. Furthermore, Adobe Flash repeatedly suffers from vulnerabilities and exploits targeting the platform. This paper reports our measurement study for a better understanding of current Flash uses. Based on a large scale evaluation of the top 1 million websites, the most detailed classification of Flash uses to date was performed. We analyzed the current uses of Flash related to user tracking, as well as to malicious files used by exploit kits.

We further introduced Garrick, a novel crawling framework to perform our measurement. Garrick is able to mimic any browser, plugin and operating system configuration so that fingerprinting scripts can be tricked to deliver malicious Flash files.

Our results show that Flash is currently used by approximately 7.5% of the top 1 million websites, with 62% of the Flash content coming from third-parties. Popular websites use Flash more often and more Flash content comes from third-parties than on other websites. From a security perspective, malicious Flash files served by highly targeted malvertising campaigns are an ongoing challenge.

Acknowledgments

The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged. Furthermore, this work has been supported by the Austrian Research Promotion Agency under grant 853264 (PriSAd), a joint project of Nimbusec GmbH and the St. Pölten University of Applied Sciences.

References

[1] The Recorded Future Blog, "Gone in a flash: Top 10 vulnerabilities used by exploit kits," November 2015, https://www.recordedfuture.com/top-vulnerabilities-2015/ [Online; Accessed on December 1, 2018].

[2] K. J. Higgins, "Adobe flash flaws dominate exploit kits in 2016," Threat Intelligence, December 2016, https://www.darkreading.com/threat-intelligence/adobe-flash-flaws-dominate-exploit-kits-in-2016/d/d-id/1327645 [Online; Accessed on December 1, 2018].

[3] Adobe Systems, "Adobe security bulletin," APSB18-44, November 2018, https://helpx.adobe.com/security/products/flash-player/apsb18-44.html#affectedversions [Online; Accessed on December 1, 2018].

[4] Redhat, "CVE-2018-15981," Red Hat CVE Database, November 2018, https://access.redhat.com/security/cve/cve-2018-15981 [Online; Accessed on December 1, 2018].

[5] Adobe Corporate Communications, "Flash & the future of interactive content," July 2017, https://blogs.adobe.com/conversations/2017/07/adobe-flash-update.html [Online; Accessed on December 1, 2018].

[6] Adobe Systems, "Adobe flash runtimes: Statistics," http://www.adobe.com/products/flashruntimes/statistics.html [Online; Accessed on December 1, 2018].

[7] BuiltWith, "Shockwave flash embed usage statistics," https://trends.builtwith.com/framework/Shockwave-Flash-Embed [Online; Accessed on December 1, 2018].

[8] W3Techs, "Usage of flash broken down by ranking," https://w3techs.com/technologies/breakdown/cp-flash/ranking [Online; Accessed on December 1, 2018].

[9] Google, "Youtube now defaults to html5," January 2015, https://youtube-eng.googleblog.com/2015/01/youtube-now-defaults-to-html5_27.html [Online; Accessed on December 1, 2018].

[10] L. Lorenzetti, "Google is ditching adobe flash advertising," February 2016, http://fortune.com/2016/02/11/google-adobe-flash-advertising/ [Online; Accessed on December 1, 2018].

[11] IAB Technology Laboratory, "Transitioning video ads from flash to html5," December 2017, https://iabtechlab.com/html5videotransition/ [Online; Accessed on December 1, 2018].

[12] J. Pudelek, "Migrating games from flash to open web standards on facebook," Facebook for Developers, July 2017, https://developers.facebook.com/blog/post/2017/07/25/Games-Migration-to-Open-Web-Standards/ [Online; Accessed on December 1, 2018].

[13] H. Shacham, "The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86)," in *Proc. of the 14th ACM Conference on Computer and Communications Security (ACM CCS'07), Alexandria, Virginia, USA*. ACM, October-November 2007, pp. 552–561.

[14] D. Blazakis, "Interpreter exploitation: Pointer inference and jit spraying," Black Hat & Defcon 2010, July 2010, http://www.semantiscope.com/research/BHDC2010/BHDC-2010-Paper.pdf/ [Online; Accessed on December 1, 2018].

[15] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster: Exploring the ecosystem of web-based device fingerprinting," in *Proc. of the 2013 IEEE Symposium on Security and Privacy (SP'13), Berkeley, California, USA*. IEEE, May 2013, pp. 541–555.

[16] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Guerses, F. Piessens, and B. Preneel, "Fpdetective: dusting the web for fingerprinters," in *Proc. of the 2013 ACM SIGSAC Conference on Computer & Communications Security (ACM CCS'13), Berlin, Germany*. ACM, November 2013, pp. 1129–1140.

[17] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The web never forgets: Persistent tracking mechanisms in the wild," in *Proc. of the 2014 ACM Conference on Computer and Communications Security (ACM CCS'14), Scottsdale, Arizona, USA*. ACM, November 2014, pp. 674–689.

[18] G. Merzdovnik, M. Huber, D. Buhov, N. Nikiforakis, S. Neuner, M. Schmiedecker, and E. Weippl, "Block me if you can: A large-scale study of tracker-blocking tools," in *Proc. of the 2017 IEEE European Symposium on Security and Privacy (EuroS&P'17), Paris, France*. IEEE, April 2017, pp. 319–333.

[19] P. Eckersley, "How unique is your web browser?" in *Proc. of the 10th International Conference on Privacy Enhancing Technologies (PETS'10), Berlin, Germany*, ser. Lecture Notes in Computer Science, vol. 6205. Springer, Berlin, Heidelberg, July 2010, pp. 1–18.

[20] B. Duncan, "Decline in rig exploit kit," June 2017, https://researchcenter.paloaltonetworks.com/2017/06/unit42-decline-rig-exploit-kit/ [Online; Accessed on December 1, 2018].

[21] C. Cimpanu, "Rig exploit kit usage declines as browsers are getting harder to hack," June 2017, https://www.bleepingcomputer.com/news/security/rig-exploit-kit-usage-declines-as-browsers-are-getting-harder-to-hack/ [Online; Accessed on December 1, 2018].

[22] S. Ford, M. Cova, C. Kruegel, and G. Vigna, "Analyzing and detecting malicious flash advertisements," in *Proc. of the 2009 Annual Computer Security Applications Conference (ACSAC'09), Honolulu, Hawaii, USA*. IEEE, December 2009, pp. 363–372.

[23] T. V. Overveldt, C. Kruegel, and G. Vigna, "Flashdetect: Actionscript 3 malware detection," in *Proc. of the 15th International Symposium on Recent Advances in Intrusion Detection (RAID'12), Amsterdam, The Netherlands*, ser. Lecture Notes in Computer Science, vol. 7462. Springer, Berlin, Heidelberg, September 2012, pp. 274–293.

[24] C. Wressnegger and K. Rieck, "Looking back on three years of flash-based malware," in *Proc. of the 10th European Workshop on Systems Security (EuroSec'17), Belgrade, Serbia*. ACM, April 2017, pp. 6:1–6:6.

[25] Z. Li, K. Zhang, Y. Xie, F. Yu, and X. Wang, "Knowing your enemy: Understanding and detecting malicious web advertising," in *Proc. of the 2012 ACM Conference on Computer and Communications Security (ACM CCS'12), Raleigh, North Carolina, USA*. ACM, October 2012, pp. 674–686.

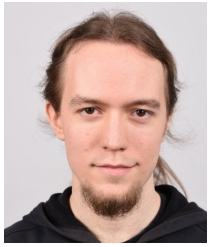
[26] A. Zarras, A. Kapravelos, G. Stringhini, T. Holz, C. Kruegel, and G. Vigna, "The dark alleys of madison avenue: Understanding malicious advertisements," in *Proc. of the 2014 Conference on Internet Measurement Conference (IMC'14), Vancouver, British Columbia, Canada*. ACM, November 2014, pp. 373–380.

[27] A. Soltani, S. Canty, Q. Mayo, L. Thomas, and C. J. Hoofnagle, "Flash cookies and privacy," in *Proc. of the 2010 AAAI Spring Symposium: Intelligent Information Privacy Management, Palo Alto, California, USA*. AAAI, March 2010, pp. 158–163.

[28] S. Englehardt and A. Narayanan, "Online tracking: A 1-million-site measurement and analysis," in *Proc. of the 2016 ACM Conference on Computer and Communications Security (ACM CCS'16), Vienna, Austria*. ACM, October 2016, pp. 1388–1401.

Author Biography

Damjan Buhov is a researcher in the Josef Ressel Center for Unified Threat Intelligence on Targeted Attacks and a lecturer at the University of Applied Sciences in St. Pölten, Austria. He received his B.Sc. and M.Sc. degrees from the Ss. Cyril and Methodius University - Skopje, Faculty of Computer Science and Engineering and is currently working towards his PhD degree. His research interests are in the areas of security & privacy in the mobile technologies as well as program analysis.



Julian Rauchberger completed a Master of Science (MSc), Information Security from the University of Applied Sciences in St. Pölten, Austria in 2018 and is currently working there as a researcher. He is particularly interested in researching the patterns of threat actors spreading malware.

Sebastian Schrittwieser heads the Institute of IT Security Research and is a professor (FH) for information security at the University of Applied Sciences St. Pölten, Austria. Furthermore, he is the head of the Josef Ressel Center for Unified Threat Intelligence on Targeted Attacks. He received a doctoral degree in informatics with focus on information security from the Vienna University of Technology in 2014. Sebastian's research interests include, among others, code obfuscation, digital forensics, and mobile security.