Anatomy of Exploit Kits[⋆]

Preliminary Analysis of Exploit Kits as Software Artefacts

Vadim Kotov and Fabio Massacci

DISI - University of Trento, Italy
surname@disi.unitn.it

Abstract. In this paper we report a preliminary analysis of the source code of over 30 different exploit kits which are the main tool behind drive-by-download attacks. The analysis shows that exploit kits make use of a very limited number of vulnerabilities and in a rather unsophisticated fashion. Their key strength is rather their ability to support "customers" in avoiding detection, monitoring traffic, and managing exploits.

Keywords: exploit kits, web threats, malware analysis.

1 Introduction

Over the last few years, the volume of web-borne malware significantly increased. According to various security reports [1,10] malicious URLs attacking browsers and their add-ons constitute the majority of all Internet threats. They exploit vulnerabilities in the web browsers and their add-ons in order to download malware executable onto the victim machine. This kind of attack is called drive-by-download [11]. In the worst cases compromised clients behind a company firewall can be used to wreak havoc on critical systems. In the best ones, they lay the basis for a large malware infrastructure that can be used for identity theft or banking fraud.

Drive-by downloads are managed by a so called *exploit kit* (or exploit pack) - a server application delivering malware instead of web content [12]. Its key feature is that in order to deploy it a "customer" of this tool does not need to be more expert in web technologies than a lousy system administrator. One only need to pay the developer of the kit for the code and possibly other services (such as obfuscation). These characteristics ultimately increase the number of possible attackers and the risks for the community at large.

1.1 Our Goals and Contribution

In this paper we explore the leaked source code for some popular exploit kits. In our analysis we pursued the following goals:

[⋆] Work partly supported by the European Union by the Erasmus Mundus Action 2 Programme and the Project EU-FP7-SEC-CP-SECONOMICS.

- Study the functional aspects of exploit kits and offer a taxonomy for the routines implemented in them;
- Classify the exploit delivery mechanisms;
- Uncover web crawler evasion techniques that are used by exploit kits.
- Understand the user interface of an exploit kit, find out what data it provides and what management capabilities are available to the customer.
- Investigate the code re-use in various exploit kits and determine if there is a common code base used by malware authors.
- Study the methods of code protection mechanisms that are aimed to prevent unauthorized code distribution and complicate the analysts' work.

The results of our study are quite surprising. We expected exploit delivery mechanisms to be sophisticated - to work as snipers, performing a careful study of the remote machine and then delivering only the right exploit to the right victim. While the study is performed by most kits, its results are not used in a significant way to select the exploit. Instead the attack is performed in machine-gun style. It seems that the main purpose of victim fingerprinting is to make statistics and "dashboards" of traffic and malware installations. In other words exploit kits' main target is to "improve the customer experience". A large number of successfull infections is expected to come by large volumes of traffic instead of sophisticated attacks.

2 Related Works

Very few papers have examined exploit kits as a class of software artefacts. Most studies on infiltrations (such as those by Savage, Paxson and their groups [4,9]) usually focus on a single tool and try to reconstruct the whole food chain from the web-user to the final bad guy monetizing the result. For example, Motoyama et al. [9] analyzed the private messages exchanged in 6 underground forums. They analyzed whether sellers did re-use the same ID, whether transactions were moderated, or reputation systems were in place. A similar study focusing on the Chinese sites has been done in [13]. Yet they did not consider analyzing the actual malware posted on those forums. Franklin et al. [4] and Herley et al. [8] have analyzed (with opposite conclusions) the whole chain for spam and malware goods distribution but have not considered the individual artefacts at the start of the chain. Grier et al. [6] described the landscape of exploit kits and malware families, with more detailed focus on the latter. Their main result is a statistical analysis that shows which exploit kits are used to distribute which malware on what kind of traffic. For example, the authors determined that modern exploit kits deliver 32 different malware families including, ZeroAccess, SpyEye and Zeus as ones of the most famous. But there were no analysis of exploit kit technologies as much.

Only the author of [7] did study exploitation capabilities of the popular malware toolkits. However, the paper only considers a small number of instances and does not provide a comparison of their features as software artefacts. Our

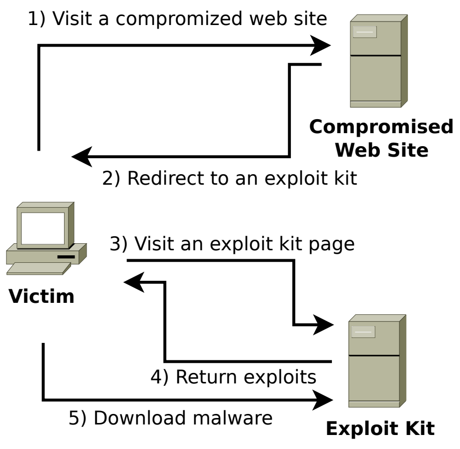

1) Visit a compromized web site
Compromised Web Site
2) Redirect to an exploit kit
3) Visit an exploit kit page
Victim
4) Return exploits
5) Download malware
Exploit Kit

Fig. 1. Scheme of drive-by-download attack

perspective is to take a wider look and investigate the structure of the crimeware packs. Another paper in which the actual instances have been considered is the work by Cova et al.[3] which focuses on fishing kits.

3 Background

An exploit kit is a software tool traded on the black market and used by cybercriminals to perform drive-by-download attacks. From an implementation point of view an exploit kit is an HTTP server-side application, that, based on request headers, returns a page with an appropriate set of exploits. Its main purpose is silently downloading and executing malware on the victim machine by taking advantage of browser vulnerabilities. Errors in applied programming interfaces or memory corruption based vulnerabilities allow an exploit to inject a set of instructions (called shellcode) into the victim process. Shellcode on its turn downloads a malware executable to the victim's hard drive and executes it. The executable that gets installed on the target system is completely independent from the exploit pack (see [6] for a distribution of malware families provisioned by the different exploit kits). An owner can "arm" it with any malicious application of her choice.

Fig. 1 depicts the generic scenario of drive-by-download attack [11]. A victim visits a compromised web site, from which she gets redirected to the exploit kit page. Various ways of redirection are possible: an <iframe> tag, a JavaScript based page redirect etc. The malicious web page then returns an HTML document, containing exploits, which are usually hidden in an obfuscated JavaScript code. If at least one exploit succeeds, then a victim gets infected. Successful exploitation means, that the shellcode injected has finished flawlessly and hence accomplished its task - to download and execute a malicious program.

How successful an exploit kit is depends on such factors as an operating system version, type and version of a browser and its add-ons, presence of security measures.

Apart from the exploits a kit has an administrative panel - a dashboard that provides statistics and allows a user to configure the tool. Even the earliest kits such as Mpack and IcePack had this feature [12].

Exploit kits are usually constructed from open-source components such as an Apache web server, a PHP server-side scripting engine and a MySQL database.

In order to protect users from drive-by-download attack, two main strategies are normally deployed:

1. Protect end users with malware scanners and other security means which intercept the malware on the fly or stop the exploit from completing;
2. Build black lists of URLs (such as those behind Google Safe Browsing). These lists are constructed by security web crawlers, which instead of indexing the site content, check the web page with malware scanners or analyze its behavior in a sandbox. In fact this can be done by the search engine's robots in addition to traditional content indexing.

These two defence mechanisms determine the presence of yet another feature of an exploit kit: detection evasion. In this sense a kit can implement the following self-protection measures:

– Code obfuscation, deployed in order to fail malware scanners' signatures and heuristics. For example the Black Hole exploit kit [5] applies a polymorphic obfuscation algorithm to its malicious JavaScript code.
– Checking itself with antiviruses to find out whether the signature for the current obfuscation scheme already exists and whether it is time to update the obfuscation algorithm.
– Restricting search robots activity by disallowing indexing policy in the "robots.txt" file.
– Mimicking an innocent web page when encountering an unsupported user agent (search robots, downloading software, etc.).
– Looking itself up in the black lists of URLs and IP addresses (like Google Safe Browsing) and, if found, rebind itself to another server/domain.

Finally, an exploit kit is a software product in itself and therefore must have some features of legitimate software such as source code protection, licensing, binding to single server/domain, etc. For example the Fragus exploit kit is protected with a commercial tool IonCube[1], which also makes it impossible to run the kit under the domain/server [2] that is different from the customer's.

To better understand the idea of exploit kit let's consider the following example: a user running Firefox 1.0.4 with Adobe Reader v.8.1.1 under the Windows XP opens a web page from a compromised server. An invisible iframe (left by the hacker) loads a page from another web server hosting the Eleonore 1.4.4mod exploit kit. On the server side a corresponding PHP script parses the client's

[1] `www.ioncube.com`, checked on 14 Aug 2012

HTTP request headers and retrieves the following information: name and version of the browser (Firefox 4.0), name and version of operating system (Windows XP). Based on that the PHP script selects the set of exploits such as one for CVE-2005-2265 vulnerability (targeting Firefox 1.0.4). The exploits selected are wrapped in the JavaScript code, which then gets obfuscated and returned to the client. If the exploit succeeds (if nothing stops it from executing), then the shellcode takes control over the browser's execution. It calls the URLDownloadToFile (Urlmon.dll) and then WinExec (Kernel32.dll) functions to download and execute an instance of Zeus trojan that is stored on the attacker's web server. Once the shellcode makes the request to the exploit kit server (to retrieve the trojan), the corresponding PHP script adds the successful exploitation record to the database and returns the contents of the binary. The owner of this malicious server in the administrative panel can see how many visitors were lured to the malicious page and how many of them were infected.

The features that we have listed above are the capabilities of an exploit kit that could be implemented. Whether they are really used in real-world tools and if yes then to what extent - is the question that we try to answer in this paper.

4 Collected Data

To collect the data for analysis two sources of information were used:

– A list of exploit packs, available at Contagio Malware Dump [2] security blog.
– Advertising and leaked code on various black hat forums.

Altogether we identified information for more than 70 exploit kits and out of those we were able to successfully deploy 33 instances of 24 families. Our semantics of successful is that the kit installs, runs, and is able to deliver a prototype malware of our choice to an appropriate client. We are now running a more sophisticated experiment in which we benchmark whether all claims about number of successful installations by the exploit kit developer in terms of successful installation are correct. The full list of deployed kits is presented in Table 2. Our collection includes the most famous products on the black market, according to the reports of Kaspersky Lab [12], Sophos [5] and Symantec [2].

Among all deployed exploit kits there is one that we can not yet fully analyse - Crimepack v.3.1.3. It was obfuscated with a powerful commercial protector named IonCube for which, to our knowledge, there is no good deobfuscation tool. But we were still able to extract some information from Crimepack using black box analysis of the deployed sample.

Figure 2 shows the connections between an exploit kit and some related entities such as the victim, the malware scanner or security crawler and the developer.

All the kits in our collection were written in PHP and were designed to work in bundle with MySQL database. They present the following key architectural components:

[2] `http://contagiodump.blogspot.it/2010/06/overview-of-exploit-packs-update.html`, checked on 14 Aug 2012.

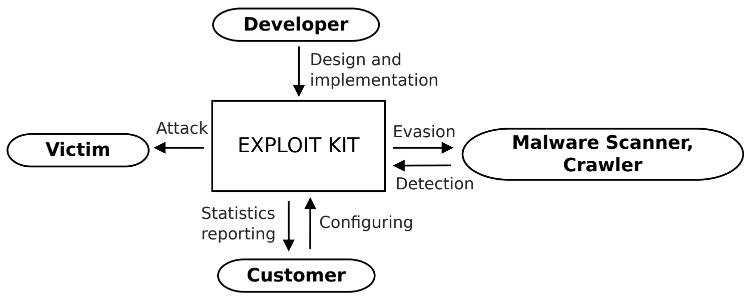


Fig. 2. Exploit kit use cases

Offensive Component which is responsible for analyzing and ultimately attacking vulnerable machines;

Defensive Component that protects the toolkit from detection by malware scanner (such as obfuscation of pages);

Management Component which supplies the reporting and configuration components of an exploit kit to support the customer;

Protection Mechanisms , which includes means of protection applied to an exploit kit for preventing unauthorized distribution and complicating the process of reverse engineering.

5 Offensive Component

To identify the operation of the offensive routes we performed code inspection, debugging and sandboxed code execution. The offensive routine consists mainly of two parts. The first one occurs on the request of the web client, when a victim gets redirected to the bad page. If the exploit is successful, the second part is activated on shellcode request after it has taken control over the target application.

The first part consists of the following steps:

1. *User agent detection* determines operating system (OS) and user agent (UA) used by the victim.
2. *IP blocking* blocks a visitor on the next visit (based on IP address).
3. *UA validation* - If the OS or the UA are not supported do either of the following actions:
   - output an innocent looking page like “Site is under construction”, and provide the response status 200 (OK);
   - redirect a visitor to another page or web site by specifying its address in the “Location” header of the server response;
   - output an error page and provide the response status of error (e.g. 404);
   - deliver anyhow some exploits in the hope that the client is vulnerable.
4. *Exploits selection* - Select the subset of exploits for the determined OS and UA or follow corresponding execution branch.

Table 1. Presence of the offensive routine steps

Step	Present (%)	Absent (%)
User agent detection	100%	0
IP blocking	79%	21%
UA Validation	88%	12%
Exploits selection	82%	15%
Exploits obfuscation	82%	18%
Executable Delivery	100%	0%

5. *Exploits obfuscation* - Obfuscate the generated malformed HTML page. By this we mean the on-line obfuscation, when the attacking page goes through some transformations that change its appearance.

The second part has only one step, which is:

1. *Delivery of malware executable* - returns the malware executable file and as a follow up updates the successful exploitation statistics.

Each exploit kit largely follows the proposed scenario, irrespective of the year of deployment: Icepack kit appeared in 2007, while Phoenix is a comparatively recent product, its 3.1 update was released in 2012. Therefore we conjecture that the functional architecture is essentially stable.

A summary of the findings is shown in Table 1. The full analysis of the server side attack scenario can be found in Table 2. The results do not sum up to 100% as in some cases it was not possible to ascertain exactly what the kit does. Out of these results we can make some conclusions:

1. 88% of exploit kits perform user agent validation, which means that if a browsing robot wants to detect an attack it must send a user agent string of a vulnerable browser (Internet Explorer 6 under Windows XP is going to work for every kit analyzed) or, on the other hand, a user can change the user agent string to an unsupported one (e.g. wget under OpenBSD) in order to "trick" the kit and avoid infection.
2. 64% of exploit kits perform both IP blocking and Exploits selection, which complicates the analysis of a kit in the wild. Offensive capabilities of an exploit kit in the wild can only be revealed from different IP addresses and using different user agent strings.
3. All exploit kits in our collection have a separate piece of code responsible for executable delivery. Thus, exploit kits can keep an accurate score of the machines that were actually infected.
4. In a surprisingly large number of cases (36%), irrespective of the result of the UA validation, the exploit kit will anyhow throw some attacks. The UA validation code does not seem to be used in a significant way to select the exploit appropriately.

In terms of vulnerability analysis the picture was surprising: among the 70+ exploit kits that we had identified only a bit more than 110 vulnerabilities are

Table 2. Full data set and offensive component analysis results

Name	Version	IP Block.	UA Detec.	UA Valid.	Follow-up	Sel.	Obf.
0x88	UNK	√	√	√	ATTACK	√	√
adpack	UNK1	√	√	√	INNOCENT	√	√
adpack	UNK2	√	√		NONE		
armitage	1.0 beta	√	√	√	INNOCENT	√	√
bleeding life	2		√	√	INNOCENT	√	
crimepack	3.1.3	√	√	√	INNOCENT	UNK	√
cry217	UNK	√	√		NONE		
eleonore	1.2	√	√	√	ATTACK	√	√
eleonore	1.4.4 mod	√	√	√	ATTACK	√	√
firepack	0.18	√	√	√	ERROR	√	√
firepack	UNK	√	√	√	INNOCENT	√	√
fragus	1.0	√	√	√	ATTACK	√	√
fragus	black	√	√	√	ATTACK	√	√
gpack	UNK	√	√	√	INNOCENT	√	√
icepack	platinum beta	√	√	√	EMPTY	√	√
icepack	platinum	√	√	√	INNOCENT	√	√
el fiesta	1.0		√	√	INNOCENT	√	√
el fiesta	1.8	√	√	√	INNOCENT	√	√
life	UNK	√	√		NONE		
mpack	0.81	√	√	√	INNOCENT	√	√
mpack	0.86	√	√	√	INNOCENT	√	√
mpack	0.91	√	√	√	INNOCENT	√	√
mpack	0.99	√	√	√	INNOCENT	√	√
mypolysploit	1.0		√	√	ATTACK	√	√
neon	UNK	√	√	√	ATTACK	√	√
nuke	UNK		√	√	INNOCENT	√	√
phoenix	2.3	√	√	√	INNOCENT	√	
rds	2.0	√	√		NONE		√
salo	UNK		√	√	ATTACK	√	
seo	UNK		√	√	INNOCENT	√	√
shaman's dream	2.0	√	√	√	ATTACK	√	√
unique	UNK	√	√	√	INNOCENT	√	√
yes	2.0		√	√	REDIRECT	√	√

Explanation of the table columns:

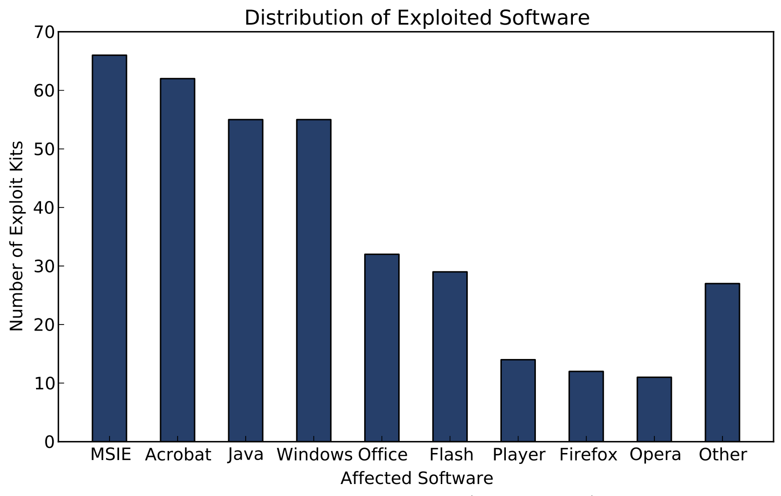
Name - name of an exploit kit;
Version - exploit kit version or UNK if we could not determine it;
IP Block. - presence if IP blocking: YES (√) or NO.
UA detec. - detection of the user agent: YES (√) or NO.
UA valid. - user agent validation, i.e. an action taken if a user agent is not supported: INNOCENT | REDIRECT | ERROR | ATTACK | NONE, where INNOCENT means an innocent looking page; REDIRECT - a redirection provided is "Location" header; ERROR - an error page; ATTACK - throw some exploits; NONE - if no action is taken.
Sel. - presence of exploit selection: YES(√) if, based on user agent information, execution path of the kit changes, otherwise NO.
Obf. - presence of exploit obfuscation: YES(√) or NO.

actually exploited. An average exploit kit had around 10 exploits ($\mu = 11.1$) which are not always fresh. Table 3 shows the mean number of exploits of certain age over the sample of 30 kits. Age of an exploit was calculated relatively to the year this kit first appeared. On average, most exploits are aimed at 1 and 2 years old vulnerabilities, which may imply that malware authors prefer to use public exploits, rather than private ones. An alternative explanation is that the time to market a reliable piece of code exploiting commodity software is significant.

The affected software is also very limited, showing a preference among exploit kit developers for easy exploits based on popular software. Figure 3 shows how many vulnerabilities affecting a given software are present in the overall sample.

Table 3. Average number of exploits by age

6 y.o.	5 y.o.	4 y.o.	3 y.o.	2 y.o.	1 y.o.	0 y.o.
0.17	1.03	1.4	2.1	2.57	3.93	1.9



The figure shows a number of exploit kits (vertical axis) that exploit particular software or class of software (horizontal axis). *Player* denotes various video/audio players (such as Real Player, Quick Time etc.), *Windows* includes exploits for components of Windows operating system, *Other* denotes various other types of exploited software (such as components of Microsoft Visual Studio, audio/video format converters, messengers etc).

Fig. 3. Preferred Software for Exploits

The entry 'Player' denotes various proprietary or open source video/audio players (such as Real Player, Quick Time etc.), while 'Windows' includes exploits for various components of Windows operating system of different versions. The category 'Other' denotes various other types of exploited software (such as components of Microsoft Visual Studio, audio/video format converters, messengers etc).

5.1 Defensive Component

Defensive means of exploit kits include IP blocking, payload obfuscation, crawlers evasion and active measures such as checking itself in various virus databases to catch the time, when it got recognized by the malware scanners in order to update the obfuscation scheme.

As we mentioned, IP blocking and obfuscation are popular measures routinely deployed in the offensive component.

Crawlers evasion can be implemented in two ways (not mutually exclusive):

1. Settings the specific indexing policy in a “robots.txt” file, which may keep search robots from collecting the information about malicious pages of an exploit kit;
2. Match a user agent string in HTTP request against known crawlers.

To find possible evasion techniques, we searched for the “robots.txt” in exploit kit files and the indexing policy it defines; we also looked for known crawler user agent strings (“Googlebot”, “Yahoo!”, “Bingbot”, “YandexBot”, etc.) from UserAgentString.com and scan through all source files for their presence. If found - analyse the context in which the string appears.

Out of this analysis we found that the large majority of exploit kits analyzed do not pay attention to crawler evasion. There are just few cases:

1. *Crimepack 3.1.3, Eleanore 1.4.4 mod* and both versions of *Fragus* have robots.txt that disallows any indexing.
2. Firepack has a list of crawlers’ names.
3. *0x88* looks for the “bot” substring in user agent name in HTTP request header.

To determine whether virus database checks were done by the exploit kit, we performed the following checks:

1. String search for the strings “virustotal” (a virus scan web service[3]) and “virtest” (a popular anonymous virus scan web service[4]), that can locate the code snippets responsible for virus database checks.
2. Looking at administrative panels of exploit kit to find the pieces of user interface that might indicate the presence of the virus database checks.

No exploit kit (except Crimepack 3.1.3 for which it is unknown) checks itself in the virus databases. However, we have not examined the source of Black Hole exploit kit. So we cannot confirm [5], which reports that it has the ability to check itself against two virus scan services.

Figure 4 shows a Venn diagram where the explicit number of items in each subset is represented by the number of crosses. Whether a kit may use or not the IP Blocking, an overwhelming majority uses obfuscation. Therefore the absence of obfuscation in a page seems a good indication that the page is unlikely to be malicious.

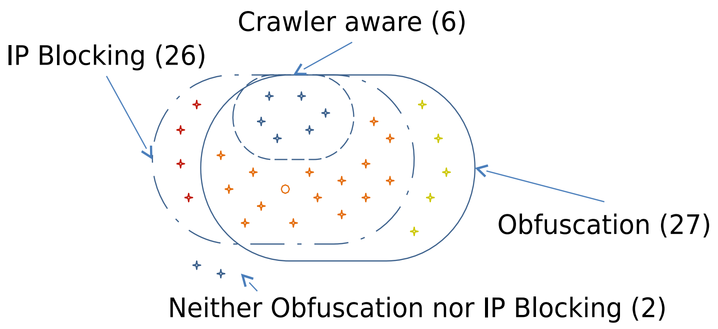
6 Management Component

The customer-oriented part of an exploit kit should provide an access to visits and exploitation statistics and offer some settings to manage the toolkit.

A step-by-step work flow can not be proposed here, because the customer handles the exploit kit by an interactive user interface. The main *use cases* are the following:

[3] `http://www.virustotal.com`, checked on 14 Aug 2012

[4] `http://www.virtest.com`, checked on 14 Aug 2012



The figure shows a Venn diagram of the defences implemented in the exploit kits analyzed. The number of items in each subset is represented by the number of crosses.

Fig. 4. Venn diagram of defensive capabilities

1. *Installation* - Install the exploit kit, i.e. allocate all resources needed for successful run.
2. *Authentication* - Authenticate exploit kit user.
3. *Control Cockpit* which normally includes
   – site visit statistics;
   – successfull exploitation statistics;
   – exploit kit settings;

Since we dealt with *leaked* code, the analysis of the installation step can only be preliminary, because automated installers might have been removed after setting up the kit. Therefore we can not say whether an automatic installer supposed to come with the software or not.

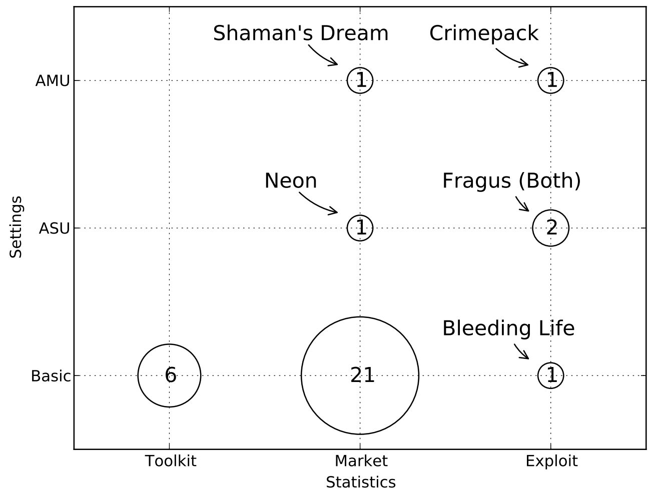
Authentication is present in every admin panel, so we do not discuss it further in detail.

The two important functions of customer oriented component are statistics reporting and settings. All kits feature some basic setting up and statistics, and a more fine grained classification is reported below:

1. *Toolkit statistics* includes the information about the exploit kit's work: total visits, exploited systems, browsers and operating systems.
2. *Market statistics* includes the information that helps a customer to manage her interaction with related markets (e.g. traffic market). The typical information of this type are referers[5] and/or countries.
3. *Exploit statistics* allows a customer to track the effectiveness of individual exploits.

We have similar three grades metric for the settings, that can be offered to the customer:

[5] A URL from which the victim came. It can be determined from the HTTP header "Referer".



The figure shows a magic quadrant depicting the exploit kits in the statistics/settings coordinate axes.
The grades for the settings axis:

Basic allows a customer to change her credentials and replace malware executable

ASU (Advanced single-user gives a customer more flexibility (manage exploits, build black lists by country, referer etc.)

AMU (Advanced multi-user) provides the settings for multi-user environment.

The grades for statistics axis:

Toolkit includes the information about the exploit kit's work: total visits, exploited systems and browsers etc.

Market statistics includes the information that helps a customer to manage her interaction with related markets (e.g. traffic market).

Exploit statistics allows a customer to track the effectiveness of individual exploits.

Fig. 5. Statistics/Settings Quadrant

1. *Basic settings* - allows a customer to change her credentials and replace malware executable.
2. *Advanced single-user settings (ASU)* - allows a customer to enable/disable exploits or/and add new ones or/and manage block lists (IP, referrers, countries) etc.
3. *Advanced multi-user settings (AMU)* - provides the settings for multi-user environment, i.e. customizable user profiles.

To perform this analysis we need to have a look at every administrative panel of our exploit kit collection and enumerate the reported statistics and the configuration options that are offered to the customer.

In Figure 5 we provide a magic quadrant depicting the exploit kits in the statistics/settings coordinate axes. The number inside the circle is a count of exploit kits that fell upon the corresponding point in the quadrant.

It can be seen that the majority of exploit kits analyzed produce advanced statistics, but offer the simplest settings possible. From statistics/settings perspective Crimepack, Fragus and Shaman's Dream are the most advanced. The conjecture is that the closer an exploit kit is to the top right corner of the quadrant the higher is its potential of supporting complex business models. In other words the advanced statistics and settings allow customers to

– keep track of the traffic they actually received, possibly to match it against what they may have bought in order to avoid waste of money in useless traffic,
– learn which exploits contribute more to the victim infecting process in order disable the bad ones (or detectable ones) and so forth.

This gives a customer a high level of flexibility in organizing the infection process and ways of its monetizing. Top right corner is the place where the Black Hole exploit kit would have been placed.

7 Code Protection

Presence of protection in an exploit kit source code can be detected based on one of the two features: (1) there is a byte code and markers of the commercial tools (Zend Guard[6] or IonCube) or (2) the code is put through some permutations and then executed. Otherwise the code is clear to read.

In summary our findings are reported below:

– *Crimepack 3.1.3* - is the only kit in our collection that uses IonCube.
– *Neon*, *Life* and *Firepack*$_{UNK}$ use Zend Guard.
– *0x88*, *Eleonore 1.4.4 mod*, *El Fiesta 1.0* and *Unique* use various ad-hoc methods of protection.
– Other kits (25) do not use any code protection.

Yet these results can not be taken as conclusive because we were dealing with *leaked* sources. For example, Fragus exploit kit, according to [2] is obfuscated with IonCube, while we obtained a clean version. All conclusions that are made from this analysis can give only general idea of the code protection within exploit kits. One of the reason the Black Hole exploit kit is not included in the study is that we could not fully restore the source code.

8 Code Re-use

Investigating the cases of code re-use could help us to better understand the production of an exploit kit. To address this question we use a token-based copy-paste detector phpcpd[7]. It reveals the snippets of repeating code among multiple PHP scripts.

[6] `http://www.zend.com/en/products/guard/`, checked on 14 Aug 2012
[7] `https://github.com/sebastianbergmann/phpcpd/`, checked on 14 Aug 2012

Table 4. Numbers and functionality of code re-use cases per pair

Kit_1	Kit_2	# of matches	Repeated Code Functionality
0x88	life	7	Admin. panel routines, obfuscation, database functions
fragus	icepack	5	Obfuscation
adpack	gpack	3	User agent detection, database functions
armitage	icepack	3	User agent detection
mpack	0x88	3	Obfuscation
rds	0x88	3	Admin. panel routines, obfuscation, database functions
eleonore	shaman	2	User agent detection, obfuscation
firepack	icepack	2	Obfuscation
nuke	seo	2	Admin. panel routines, user agent detection
yes	icepack	2	User agent detection
lefiesta	fragus	1	Obfuscation
neon	armitage	1	Array of countries names
salo	adpack	1	Obfuscated array of countries names

The detected re-used code can be divided into three groups:

1. the code repeated among different versions of the same exploit kit;
2. the code, that corresponds to open source libraries, that are used by different kits;
3. the code repeated in different *families* of exploit kits.

The first group of code is not interesting because the results of the analysis were predictable. The kits of the same family have a lot of common code.

In the second group the only open source PHP library that we have found in our collection was Geo IP[8] which allows to determine the country by IP address. This library is frequently used in exploit kits to provide country related statistics.

The third group of repeated code snippets consists of the those appearing in the different exploit kit families. The summary of code re-use cases between *different families of exploit kits* is shown in Table 4.

The highest volumes of "copy-paste" can be found at (0x88, life) and (fragus, icepack) pairs. Interestingly there is a code obfuscation algorithm that was implemented in 5 different kits (Mpack, Fragus, RDS, Life and 0x88).

In our collection there are total of 24 exploit kit families, since the number of possible pairs is $C_{24}^2 = \frac{24!}{2!(24-2)!} = 276$ and the number of pairs with at least one repeated snippet is 13, then the rate of code re-use (based on token analysis) is $13/276 = 0.047$. This means that based on analysis of the PHP language tokens similarity *there is no common code base that is used by the malware authors.* We can use the above observation to conclude that the market is also fragmented with multiple kit providers.

9 Conclusion

In this paper we have reported the first analysis of exploit kits as software artefacts. We have collected information on 70+ popular tools for malware

[8] `http://php.net/manual/en/book.geoip.php`, checked on 14 Aug 2012

distribution. Out of those we have been able to successfully deploy and test 30+ kits. They have been further analyzed.

In order to understand the nature of the class we have collected a set of leaked source code files of previously private exploit kits. Each of them we have tried to deploy, and those that we succeeded to run were selected for analysis. We used a combination of analysis techniques such as static and dynamic reverse engineering.

The result of the work can be summarized as follows:

– Exploit kits have very similar functionality, largely following the work flow described in this paper. The victim is fingerprinted (user agent and operating system information together with IP address are collected partly for exploit selection and partly for statistical purposed), a set of moderately old exploits is provisioned within an obfuscated web page in order to download and execute a malware program. Very few vulnerabilities are exploited.
– Exploit kits use IP blocking, user agent validation and code obfuscation. Very rarely they try to evade web crawlers.
– Most exploit kits provide customers with statistics and settings, allowing them to customize a toolkit and track its activity. A customer can use them to interact with other types of black markets - traffic, malware executables, hosting services, etc.
– Results of token based copy-paste analysis show that the kits analyzed seem to be written mostly independently one from another, without a common code base.
– Some exploit kits use commercial code protection (e.g. Crimepack), which means that malware authors expect to get significant amounts of money from the crimeware toolkit sales.

We expect the exploit kit technology to evolve further in direction of detection evasion and enhancement of the customer's experience. The evidence of this we can see already in the latest kits, such as Black Hole v.2. The little emphasis on exploit delivery seems to imply that a better protection can stem from large scale detection rather than individual protection.

References

1. Internet security threat report (April 2012), `http://www.symantec.com/threatreport` (Checked on September 10, 2012)
2. Coogan, P.: Fragus exploit kit changes the business model (February 2010), `http://www.symantec.com/connect/blogs/fragus-exploit-kit-changes-business-model` (Checked on September 10, 2012)
3. Cova, M., Kruegel, C., Vigna, G.: There is no free phish: an analysis of 'free' and live phishing kits. In: Proceedings of WOOT 2008, pp. 4:1–4:8 (2008)
4. Franklin, J., Paxson, V., Perrig, A., Savage, S.: An inquiry into the nature and causes of the wealth of internet miscreants. In: Proceedings of CCS 2007, pp. 375–388 (2007)

5. Fraser, H.: Exploring black hole exploit kit (March 2012), `http://nakedsecurity.sophos.com/exploring-the-blackhole-exploit-kit` (Checked on September 10, 2012)
6. Grier, C., Ballard, L., Caballero, J., Chachra, N., Dietrich, C.J., Levchenko, K., Mavrommatis, P., McCoy, D., Nappa, A., Pitsillidis, A., Provos, N., Rafique, M.Z., Rajab, M.A., Rossow, C., Thomas, K., Paxson, V., Savage, S., Voelker, G.M.: Manufacturing compromise: the emergence of exploit-as-a-service. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS 2012, pp. 821–832. ACM, New York (2012)
7. Guido, D.: A case study of intelligence-driven defense. IEEE Security Privacy 9(6), 67–70 (2011)
8. Herley, C., Florencio, D.: Nobody sells gold for the price of silver: Dishonesty, uncertainty and the underground economy. In: Economics of Information Security and Privacy (2010)
9. Motoyama, M., McCoy, D., Savage, S., Voelker, G.M.: An analysis of underground forums. In: Proceedings of ICM 2011 (2011)
10. Namestnikov, Y.: IT threat evolution: Q1 2012 (May 2012), `http://www.securelist.com/en/analysis/204792231/IT_Threat_Evolution_Q1_2012` (Checked on September 10, 2012)
11. Naranie, R.: Drive-by downloads. The web under siege (April 2009) (Checked on September 10, 2012)
12. Preuss, M., Diaz, V.: Exploit kits - a different view (February 2011), `http://www.securelist.com/en/analysis/204792160/Exploit_Kits_A_Different_View`(Checked on September 10, 2012)
13. Zhuge, J., Holz, T., Song, C., Guo, J., Han, X., Zou, W.: Studying malicious websites and the underground economy on the chinese web. In: Proceedings of MIRES, pp. 225–244 (2009)