SEPTEMBER 2013

virus BULLETIN

Covering the global threat landscape

CONTENTS

IN THIS ISSUE

JAVA FANS

All kinds of amazing things can be done in JavaScript, especially when the size is constrained. However, when you take size-optimization techniques, combine them with structure and variable-name obfuscations and cram in every malicious action that comes to mind, then you end up with something that looks like JS/Proslikefan. Peter Ferrie has the details.

page 10

TIP OF THE ICEBERG

Despite recent declines, spam still accounts for more than 70% of all email sent. Why does this happen? He Xu exposes the tip of the iceberg by analysing a recent spambot which is driven by the Andromeda botnet: Win32/Nedsym.G.

page 17

MOTIVATION FASCINATION

John Aycock looks at academia's fascination with hacker motivation and some of the papers that abound on the topic.

page 23

ISSN 1749-7027

vb

'Government-sponsored efforts to improve cybersecurity are underway ... but will they accomplish their goals?'

Stephen Cobb, ESET

IS CYBERSECURITY BY FIAT DOA?

Government-sponsored efforts to improve cybersecurity are currently underway in several parts of the world, including the USA, the UK, and the EU, but will they accomplish their goals? The answer has serious implications for many groups of people, from security practitioners to taxpayers, CIOs and CISOs, intelligence agencies and the military. Depending on your perspective, not all of the implications are positive.

I recently participated in the latest American endeavour to secure all things cyber and critical by attending the Third Cybersecurity Framework Workshop, organized by the National Institute of Standards and Technology (NIST). As you may know, something called Executive Order 13636 directed NIST to 'work with stakeholders to develop a voluntary framework for reducing cyber risks to critical infrastructure'.

I respect NIST as one of the rare government agencies which, like the Federal Trade Commission, just seems to get on with doing useful things, including the distribution of useful information (notably the Special Publication 800 series[1]). A lesser agency might have balked when asked to create a cybersecurity framework 'in an open manner with input from stakeholders in industry, academia and government, including a public review and comment process, workshops and other means of engagement'. But so far, NIST seems to be rising to that challenge.

[1] http://csrc.nist.gov/publications/PubsSPs.html.

At the workshop I attended, over 300 people were spun out into eight working groups, led by a team of facilitators who did a great job of taking input from all sides. The starting point was a draft outline of the framework[2], based on the two previous workshops. As we evaluated the work so far, there was a lot of learned and considered discussion, but one point of friction did emerge: fear that this voluntary framework, once completed and approved, will become a stick to beat companies into compliance. Might a law be passed to punish companies that do not comply with the framework? The folks from NIST insisted they had no interest in seeing this happen, but some attendees eyed the Department of Homeland Security contingent with suspicion.

And that brings us to malware. It might seem like a stretch, but please bear with me and turn to the Code of Federal Regulations 45 CFR 164.308(a)(5)(ii)(B). This is the Health Insurance Portability and Accountability Act (HIPAA) security rule that states that a Covered Entity must implement 'Procedures for guarding against, detecting and reporting malicious software'. For years now, compliance with this rule has been the law in the USA, enforced with financial penalties running into millions of dollars. Now turn to page 16 of the *Ponemon Institute*'s Third Annual Benchmark Study on Patient Privacy & Data Security[3]. Larry Ponemon's team conducted 324 interviews and compiled stats on 80 healthcare organizations.

When the results of the study were published last year, the headline was that 94% of healthcare organizations had experienced at least one data breach in the past two years, and 45% reported more than five incidents in that period. Figure 13 in the report ('Measures to ensure devices are secure enough to connect to the network') shows that a staggering 46% of healthcare organizations don't engage in any of seven listed measures to protect critical systems. Only 23% insist on having anti-malware on mobile devices that connect to the network, and only 21% scan devices for malware prior to connection. Sadly, there are many more data points beyond the *Ponemon* study[4].

For me, this all adds up to a strong case for saying that you can't legislate security. A voluntary framework might help, but as several of my fellow attendees at the NIST workshop pointed out: information security requires serious will power and commitment. Take that away, and regulation is apt to do more harm than good.

[2] http://www.nist.gov/itl/csd/cybersecurity-070213.cfm.
[3] http://www.ponemon.org/blog/third-annual-patient-privacy-data-security-study-released.
[4] http://www.technologyreview.com/news/429616/computer-viruses-are-rampant-on-medical-devices-in-hospitals/.

NEWS

UK GETS CYBER ACADEMY

The UK's information security skills will get a boost from the start of this month, as 2 September sees the opening of the nation's first Cyber Academy.

The Academy, launched by e-skills UK – a non-profit organization tasked with monitoring and improving Britain's technology skills – aims to help the nation develop its cybersecurity skills using a collaborative approach that involves industry, education and government.

The Academy is supported by government investment from the UK Commission for Employment and Skills (UKCES) as well as industry partners including specialist SMEs, global systems integrators, defence leaders and businesses in sectors such as retail and finance.

This autumn will see the launch of the 'Secure Futures' campaign, aimed at encouraging youngsters to consider a career in cybersecurity, while in another bid to encourage the younger generation into the field (only 7% of information security professionals are under the age of 29), e-skills UK is developing the first nationally available degree-level apprenticeships in cybersecurity.

More details are at http://www.itskillsacademy.ac.uk/cyberacademy/.

TAIWAN GETS FREE MALWARE DATABASE

A free, publicly available malware database has been launched by Taiwan's National Centre for High-Performance Computing (NCHC) in a bid to help businesses, academics and researchers boost the nation's cybersecurity.

Taiwan suffers some 3.4 million attacks each day, and is one of the world's top sources of attack traffic (ranking seventh in the world in terms of sources of global attack traffic in *Akamai*'s State of the Internet report for the first quarter of 2013).

The Malware Knowledge Base already contains around 200,000 malware samples, and more than 1,000 are being added each month. Businesses, academics and ordinary citizens are invited to apply for access to the database via the Malware Knowledge Base website.

SYRIA SEEKS ETHICAL HACKERS

Improving cybersecurity is high on the agenda of many nations, and it appears that Syria is no exception. Late last month blogger Jeffrey Carr highlighted the fact that Syria's Ministry of Communications and Technology website is soliciting experts in the areas of ethical hacking, computer forensics, incident response and malware analysis. Carr questioned just how 'ethical' Syria's ethical hackers might turn out to be.

Prevalence Table – July 2013[1]

Malware	Type	%
Adware-misc	Adware	11.52%
Java-Exploit	Exploit	9.01%
Autorun	Worm	5.77%
BHO/Toolbar-misc	Adware	4.55%
Conficker/Downadup	Worm	3.75%
Heuristic/generic	Trojan	3.65%
Crypt/Kryptik	Trojan	3.43%
Heuristic/generic	Virus/worm	3.24%
Downloader-misc	Trojan	2.76%
Iframe-Exploit	Exploit	2.68%
Dorkbot	Worm	2.27%
Sality	Virus	2.17%
Bundpil	Worm	1.91%
Sirefef	Trojan	1.88%
Heuristic/generic	Misc	1.87%
Crack/Keygen	PU	1.76%
Injector	Trojan	1.61%
Agent	Trojan	1.53%
BitcoinMiner	PU	1.48%
Potentially Unwanted-misc	PU	1.47%
LNK-Exploit	Exploit	1.45%
Wintrim	Trojan	1.42%
Gamarue	Worm	1.34%
bProtector	Adware	1.18%
Ramnit	Trojan	1.12%
Zbot	Trojan	1.11%
Virut	Virus	1.10%
Yontoo	Adware	0.98%
Brontok/Rontokbro	Worm	0.94%
Somoto	Adware	0.92%
Lollipop/MultiBundle	Adware	0.91%
Encrypted/Obfuscated	Misc	0.90%
Others[2]		18.33%
Total		100.00%

[1]Figures compiled from desktop-level detections.

[2]Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

MALWARE ANALYSIS 1

STYX EXPLOIT PACK: INSIDIOUS DESIGN

Aditya K. Sood & Richard J. Enbody
Michigan State University, USA

Rohit Bansal
Independent Security Researcher, India

In this paper, we discuss the details and design of the Styx exploit pack.

According to the dictionary, Styx is a river in the underworld, over which Charon ferried the souls of the dead. According to the Styx service provider website, 'Styx is a river in Greek mythology that formed the boundary between earth and the underworld... It circles the underworld nine times.' So it seems that the origin of the name is as rigorous as the exploit pack itself.

The Styx exploit pack was originally marketed and sold via Styx-crypt.com (see Figure 1), the website of a Russian organization that provided obfuscation services for mangling and morphing the structure of different file formats. A couple of months ago, however, the exploit pack was removed and it is now sold on the very lucrative underground market. It has been used on a large scale thanks to its efficient design, built-in exploit obfuscation and other features.

COMMUNICATION DESIGN

Styx implements a well-defined API construct to communicate with its controller application. The use of API-based web communication procedures makes the exploit pack robust and flexible. It uses JSON and XML format for sending and receiving data. Let's look at how the target URL is constructed and how communication is achieved.

Typically, a Styx URL is constructed in the format:

http://<hostname>/<api-folder>/[command|method]

The 'hostname' is the address of the target domain. The 'api-folder' is the directory on the server that is accessed using an API key. The key is sent as a part of the HTTP request to enable authentication in order to process the command or method sent by the client. Primarily, the client has to send 'X-APIKey' in the HTTP header in order to access the API so that the server will accept the requests and sends responses accordingly. For example, Listing 1 shows an HTTP request sent by the client in order to get a list of domain names configured on the server.

Styx also implements a well-defined error-handling interface for JSON and XML-based communication models, as presented in Listing 2.

The commands used by Styx are shown in Table 1.

A number of metrics are used by Styx to determine the infection success rate and to build statistics accordingly. By default, the exploit pack has an interval of 15 seconds in real time to receive data from the client. In other words, infected machines transmit data every 15 seconds. The different metrics that are used for traffic flow analysis are as follows:

- Current Loaded – number of active infections
- Current Uniques – number of unique infections

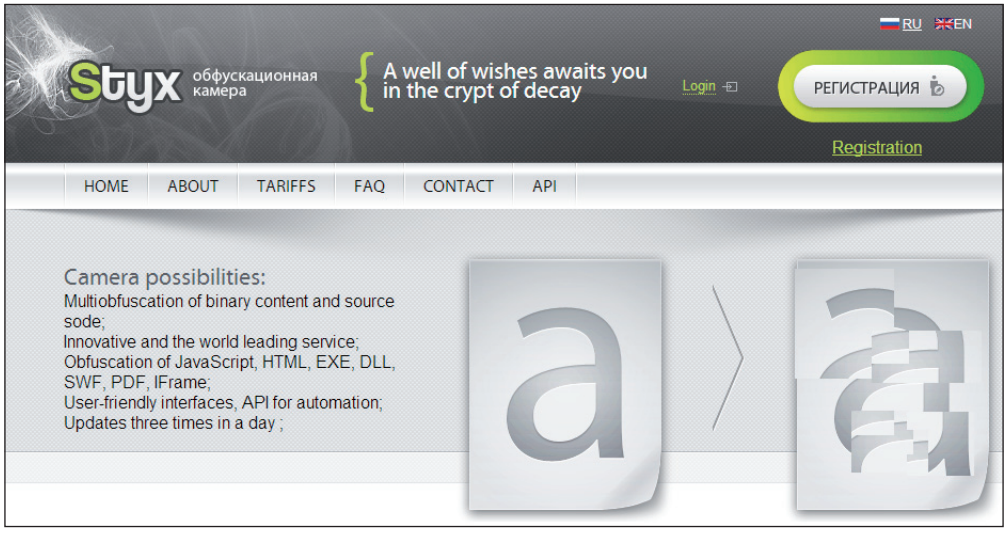


Figure 1: Original Styx service provider.

```
# Getting domain names
POST http://<styx_domain>:8888/api/getdomains HTTP/1.1
Host: <styx_domain>
Accept: application/json
X-APIKey: g48XBmTJM4Jf6LpjevOrMgXEZlRNmRluKigcx2L0UlfOYv14SEjuL81AjGdxnoR1

#Adding domain names
POST http://<styx_domain>:8888/api/adddomain HTTP/1.1
Host: <styx_domain>
Accept: application/json
X-APIKey: g48XBmTJM4Jf6LpjevOrMgXEZlRNmRluKigcx2L0UlfOYv14SEjuL81AjGdxnoR1

domain=
```

Listing 1: HTTP POST request with API key.

```
# JSON Error Flow
{
    "error": true,
    "message": "error message",
    "data": null
}

# XML Error Flow
<?xml version="1.0" encoding="utf-8"?>
<response>
    <error>1</error>
    <message>error message</message>
</response>
```

Listing 2: JSON/XML error-handling response.

Commands	Details
/api/clearSubaccStats	Clear all statistics data of a sub-account
/api/getMagicURL	Return magic API key used by sub-account for execution of commands
/api/uploadfile	Upload file
/api/getfileCheck	Check assigned file against detection
/api/getdomains	Get a list of configured domains
/api/adddomain	Add a new domain to the list
/api/createDomainSet	Create a new domain set of selection and rotation
/api/addDomainsToSet	Add domains to create a set
/api/deldomain	Remove a domain
/api/getDomainCheck	Check domain against Ghost Busters
/api/stats_global	Get global statistics by date
/api/stats_browser_n_os	Get global statistics by operating system and browser
/api/stats_country	Get global statistics by country
/api/getCurrentHitPercent	Return current and active hits
/api/getCurrentFlow	Return current data flow from the exploit pack
/api/setNotification	Set notification messages
/api/detBlockWithoutReferrer	Block access without referrer
/api/setBlockUniqueReferrers	Block (first three) access with unique referrer
/api/setBlockRepeatForIP	Block repeat access for specific IPs for hours
/api/setUsePluginDetect	Block access based on user-agent strings

Table 1: Commands used by Styx exploit pack.

- Current Hit – total number of hits
- Current Refuse – total number of IP addresses that are refused to serve exploits
- Top-5 Browsers – top five exploited browsers
- Top-5 OS – top five infected hosts
- Top-5 Countries – top five countries with the highest number of infections
- Top-5 Referrers – top five referrers, based on which exploits are served.

Styx can easily be integrated with Sutra, a traffic distribution system (TDS), to manage and build statistics regarding successful (or unsuccessful) infections based on their geographical locations.

SERVICES

Styx uses three different types of service for various functionalities. The services are discussed below.

Ghost Busters

The Ghost Busters service [1] is designed to provide flexibility in checking and verifying known domain names against active blacklists to determine whether the domain has been marked as malicious. Active domains are not mapped to any entries present in the blacklist and thus cannot be traced easily. As a result, the incoming traffic from infected systems remains active and malicious domains continue to spread malware. This prevents traffic loss. Listing 3 shows how Styx implements the domain verification check.

Ghost Busters provides a well-defined API that can be integrated into the Command & Control (C&C) panels of different automated exploit and malware infection frameworks to provide a built-in defence. The Ghost Busters system provides real-time updates on the fly, which are very beneficial for attackers in preventing the fingerprinting of domains. The Ghost Busters service also implements a robust multi-threading system to address multiple requests made at the same time. It usually takes three seconds to provide domain verification results. Figure 2 shows the Ghost Busters website.

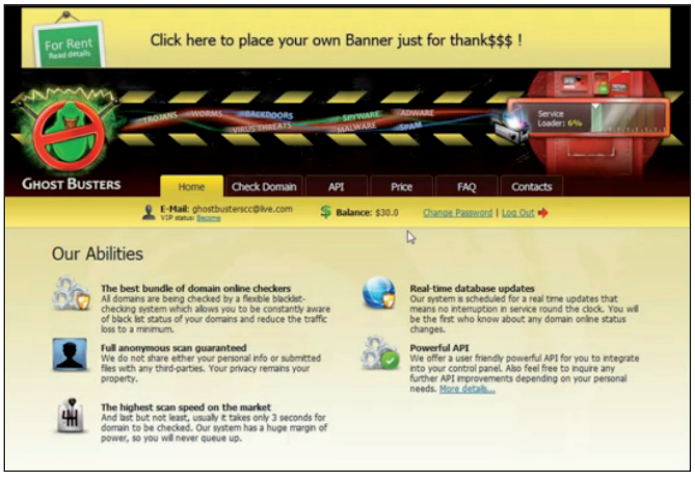


Figure 2: Ghost Busters service.

Captain Checker

The Captain Checker service is used by Styx to check whether a generated file will execute properly. Captain Checker verifies that the file is not easily detectable by the anti-protection solutions running on the end-user machines. The idea is to check whether the malicious file survives after a number of aggressive tests against known anti-virus solutions. Listing 4 shows how a simple check is performed by Styx when a malicious executable is generated.

Styx obfuscator

Styx also uses a built-in service for morphing and obfuscation. Every single exploit code served by Styx

```
// Check domain with Ghost Busters

$domain = "my-domain.com"
If (false === ($result = $api -> getDomainCheck($domain))) {
            trigger_error($api -> getErrorMessage());
} elseif ($result -> messame == 'OK')  {
           printf("your domain %s is OK, Ghostbusters said.", $domain);
} elseif {
            printf("Domain id +NOT+ clean, bro. Here is your check: %s, your domain: %s", $result ->data->public_
url, $domain);
}
```

Listing 3: Ghost Busters domain verification check.

```
// Check domain with Ghost Busters

$domain = "my-domain.com"
If (false === ($result = $api -> getFileCheck( ))) {
      trigger_error($api -> getErrorMessage());
} elseif ($result -> message == 'OK')  {
      echo "File checked. It's OK.";
} else
       printf("Another proble with your file, my
Lord. Captain Checker says it's NOT ok: %s", $result
->data->public_url);
}
```

Listing 4: Captain Checker file screening.

is properly obfuscated with this cryptor service. This substantially complicates the process of unwrapping exploit code for analysis.

FILTERS AND ACCESS RESTRICTIONS

Styx implements a number of different filters to restrict the incoming flow of unauthorized traffic. This functionality protects the exploit pack against being traced. The different sets of filters are discussed below:

- Block access without referrer: if the incoming HTTP request does not have the appropriate referrer header set, Styx blocks the request. This means that some type of referrer validation exists in the Styx exploit pack.
- Block access (first 3) with unique referrer: access to Styx web pages is blocked if the incoming requests have unique referrers. This filter is created to trigger ambiguity in accessing the Styx exploit pack.
- Block repetitive access: if the incoming requests are repetitive and originate from the same IP addresses, access is blocked immediately for an hour. This duration can be extended as required. This filter is designed specifically for scenarios in which security researchers and analysts use emulated systems to download malware.
- Filter IP addresses: the IP addresses of the infected machines that are connected to the Styx exploit pack are filtered. This is to restrict the bot traffic originating from already compromised systems.
- Filter non-*Windows* traffic: the user-agent string that accompanies incoming HTTP requests is scanned. This testing is performed to detect whether the traffic originates from e.g. *Windows* systems or mobile platforms. This option restricts the serving of the exploit in a non-reliable environment. For example, an exploit that runs on *Windows* will fail on the *Linux* platform, so with the use of this filter, traffic screening can be performed.
- Filter bots by user agent: in this filter, the incoming HTTP traffic is scanned based on user-agent strings carrying information about the crawlers and traffic collector bots. This is done to avoid automated crawling for Styx and to restrict the listing in search engines.

Once the filter is in place, the next step is to take action when the filter finds the traffic. Styx triggers three different actions by replying with one of the following:

- 402 Payment required
- 404 Page not found
- Redirect to BackURL – 302.

EXPLOIT DISTRIBUTION AND ANALYSIS

Now let's look at exactly how Styx downloads malware onto users' systems. In a number of deployments, Styx uses multiple iframe redirectors to redirect browsers to a malicious domain. For example, the typical URLs used by Styx are shown in Listing 5. The random strings are actual API keys that authenticate the client HTTP requests to the server.

On successful redirection to a malicious domain, the browser sends a GET request to download a malicious file (in this example, it is Java), which exploits the vulnerability in the browser to fetch the malware. Primarily, Styx uses the PluginDetect script to map the number of vulnerable plug-ins running in the system. When an iframe is executed, the browser is redirected to the malicious domain which triggers the PluginDetect script. If plug-ins are found to be vulnerable, a requisite exploit file is served, as shown in

```
h___p://loadcontent.zapto.org:8888/jyfGy80g7h70DI9M0JzPI0osnR0839G0eQ4V0V3XG0E1oJ0Ruqs0eo9X0KMdJ12ybd/

h___p://loadcontent.zapto.org:8888/zRu1S80FSmy0vSvg0vOqU0nVcA16fx70NXCG0IZJv0djlf0H7Tt06qeU0BKhn06ys0/

http://getstatlink.com/m2DM610qtKM0iVWv0iKBR0O75g0PSu00DBlZ0Xzlz0ixge0xxwL06Yex0FsBj0K4wd0d5AJ0iROl/

http://getstatlink.com/m2DM610qtKM0iVWv0iKBR0O75g0PSu00DBlZ0Xzlz0ixge0xxwL06Yex0FsBj0K4wd0d5AJ0iROl/mCYoHHs.js
```

Listing 5: Styx exploit pack – URL design.

Figure 3: Malicious Jar file used by Styx.

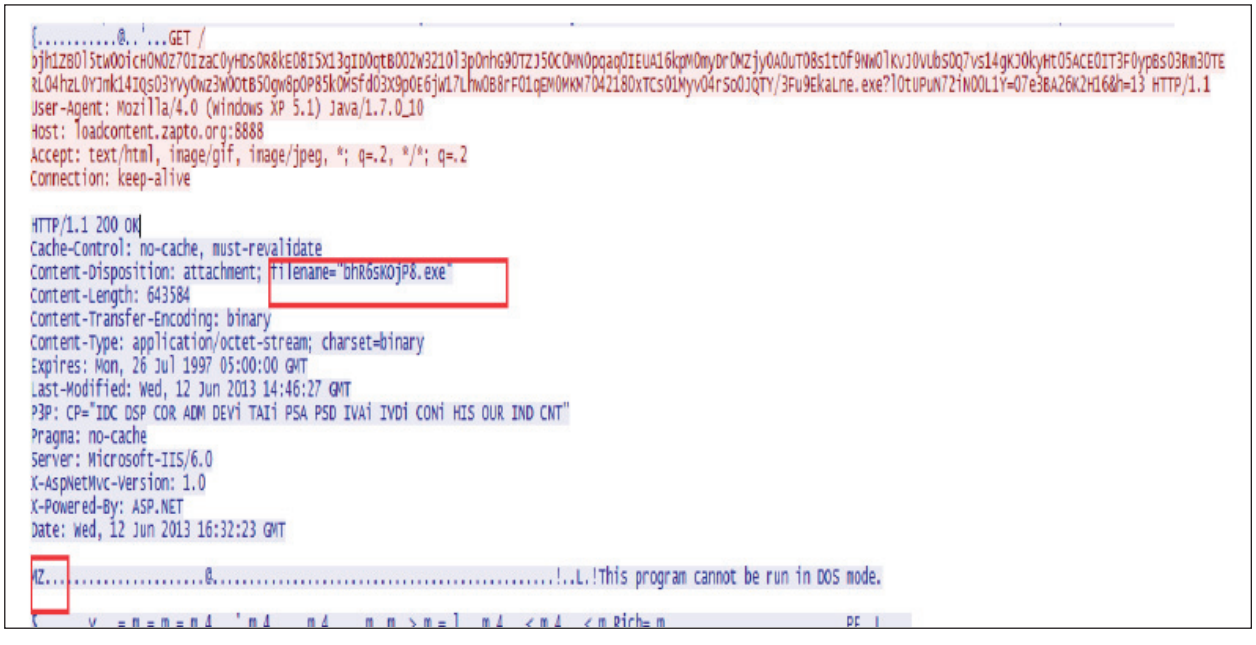


Figure 4: Downloading malicious executable.

Figure 3. If there are no vulnerable plug-ins, the malicious domain either serves no HTTP response or redirects the browser to a legitimate domain such as the *Google* search engine.

On successful exploitation, Styx serves the malicious executable, as shown in Figure 4.

Once the malware is served and successfully installed, it connects back to the Styx exploit pack administration panel to send a notification about the installation and to update the statistics, as shown in Figure 5. As one can see, the bot is sending random numbers as a part of the ps0 parameter. There is a possibility that the C&C panels used by the botnet and exploit packs such as Styx are hosted on the same domain. In certain scenarios, to increase security, the malware authors use two different domains for the exploit pack and the botnet C&C panel.

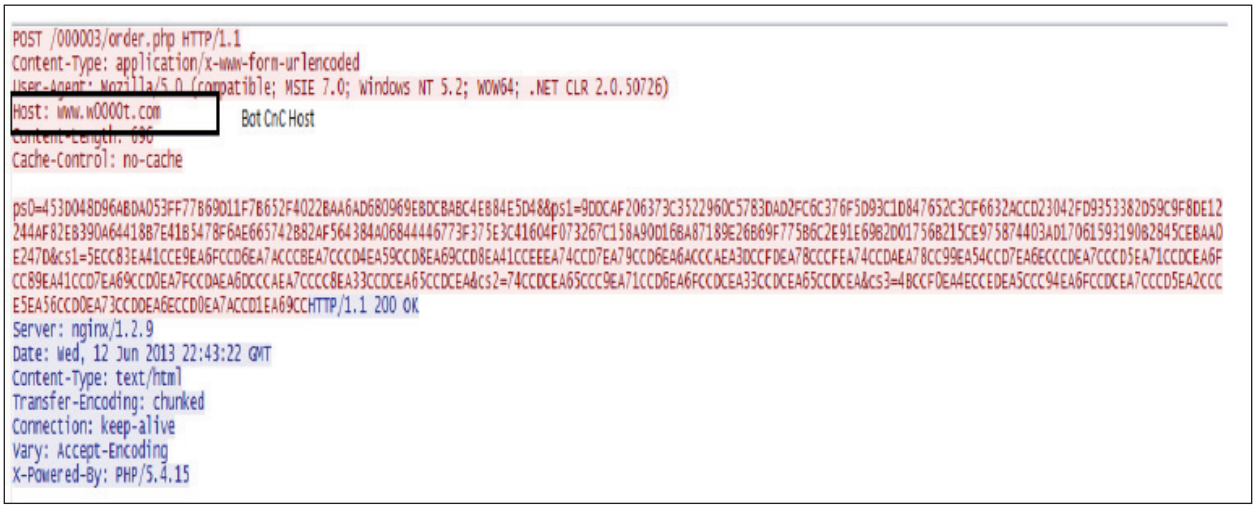


Figure 5: Bot (malware) communicating with C&C.

```
alert tcp $HOME_NET 1024: -> $EXTERNAL_NET any (msg:"Win32.Exploit.Styx -  CnC Communication"; flow:
established,to_server;
urilen:>200;
content:"GET ";
depth:4;
content:".exe?";
distance:200;
within:100;
content:"=";
within:30;
content:"|26|h=";
within:30;
fast_pattern;
content:"User-Agent: Mozilla/4.0 (Win";
distance:0;
content:!"|0d 0a|Cookie|3a| ";
reference:md5,d5cc74e25577706982a71eb4acbfadc1; pcre:"/\.exe\?[\w]+=[\w]+&h=[\d]{1,2}\x20HTTP\/1\.1/";
classtype:ExploitKit;
sid:XXXXXXXXX; rev:1; )
```

Listing 6: Styx exploit pack signature.

Styx uses CVE-2013-0422 [2] on a large scale to infect end-user machines by exploiting vulnerable installations of Java code. For constructing payloads and applets for Java exploitation, Styx inherits the power of the Java Network Language Protocol (JNLP) for running Java code outside the browser as a standalone application.

DETECTING STYX EXPLOIT PACK

Based on Styx functionality, we have written a Snort signature (presented in Listing 6) which can be used to trace malicious traffic generated by the Styx exploit pack in the wild.

FURTHER READING

Other researchers have blogged about the Styx exploit pack's infection mechanisms. To understand how Styx serves an exploit, an interesting case study has been discussed in [2, 6]. General information about the features and characteristics of the Styx exploit pack have been

presented in [3] to show the advancements in code and working. A list of simple detection patterns has been presented in [4] so that appropriate signatures can be designed to detect the Styx exploit pack. A comparison report [5] of the Styx exploit pack with other existing browser exploit frameworks clarifies the ongoing state of exploit packs. Finally, a general exploit distribution mechanism used by the Styx exploit pack covering a real-time case study is presented in [8].

CONCLUSION

This paper dissects the design and behaviour of the Styx exploit pack in detail. The complete design analysis will help researchers and analysts to understand more about the different elements of the Styx exploit pack. We hope that these kinds of analytical details will help the security community to build more robust protection solutions to subvert the infections spread by automated exploit packs such as Styx.

REFERENCES

[1] Ghost Busters. http://www.youtube.com/watch?v=mqWILzUnsmw.

[2] CVE-2013-0422. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-0422.

[3] The infection of Styx Exploit Kit (Landing page: painterinvoice.ru + Payload: PWS/Ursnif Variant). February 2013. http://malwaremustdie.blogspot.co.uk/2013/02/the-infection-of-styx-exploit-kit.html.

[4] Styx Exploit Kit Analysis – building a bridge to the underworld. April 2013. http://malforsec.blogspot.co.uk/2013/04/styx-exploit-kit-analysis-building.html.

[5] Inside Styx Sploitpack 4.0 – Exploit Kit Control Panel. May 2013. http://malware.dontneedcoffee.com/2013/05/inside-styx-2013-05.html.

[6] Styx Exploit Kit. December 2012. http://www.malwaresigs.com/2012/12/19/Styx-exploit-kit/.

[7] An Overview of Exploit Packs (Update 19.1). April 2013. http://contagiodump.blogspot.com/2010/06/overview-of-exploit-packs-update.html.

[8] Surgihalli, S.; Krishnasamy, V. Styx Exploit Kit Takes Advantage of Vulnerabilities. June 2013. http://blogs.mcafee.com/mcafee-labs/Styx-exploit-kit-takes-advantage-of-vulnerabilities.

MALWARE ANALYSIS 2

FANS LIKE PRO, TOO

Peter Ferrie
Microsoft, USA

There are all kinds of amazing things that can be done in JavaScript, especially when the size is constrained, such as playing the 1KB game 'Mine[love]craft'. However, when you take the size-optimization techniques from there, combine them with structure and variable-name obfuscations, cram in every malicious action that comes to mind and, of course, have no limit on the file size, then you can end up with something that looks like JS/Proslikefan.

WMF-WTF?-GQ

The virus begins as a wall of text, using no unnecessary whitespace (so the entire script is a single line of nearly 46KB characters in length). It uses random-looking variable names that are all eight characters long (or seven characters, for particular objects) and which differ only in the fifth and sixth characters (or just the fifth character for the seven-character version), making it difficult to tell them apart. As a result, we end up with lines like 'wmfyefgq+wmfywgq[90]+wmfyipgq+wmfygpgq(wmfyrsgq(wmfyoegq,wmfybjgq))+wmfykigq' (quick, how many unique variables are there?).

The virus uses other size optimizations, such as '!0' to replace 'true' and '!1' to replace 'false', exponent form instead of large numbers (e.g. 36e5 instead of 3600000 to represent one hour), and avoids semicolons as much as possible by using commas instead. The use of commas even extends to the return statement, where the virus places multiple assignment lines prior to the actual return value. One thing to note, though, is that every line has a purpose. There are no garbage instructions in the code at all. The obfuscation is strictly to make the reading difficult, rather than to mislead the reader.

The code begins like this:

```
(function(wmfyddgq,wmfynygq){wmfyqqgq="",...})(funct
ion(){return window},function(wmfyivgq){...}),functi
on(wmfydvgq,...){wmfyilgq=...}(...,function(wmfygzgq
){...},...);
```

This can be 'simplified' to

```
(function(){})(),function(){}();
```

The line declares two anonymous functions, and invokes first the left one and then the right one. The first function is declared as accepting two parameters, which are defined during the invocation. The parameters are both anonymous functions, too. The first parameter function returns the name of an object ('window'). The second parameter function

accepts one string parameter and splits the string into an array of its individual characters (not shown).

WINDOW OF OPPORTUNITY

The virus executes the first parameter function, and attempts to access the 'window' object. The access is performed inside a protected block so that the virus can intercept any error that occurs. The virus is expecting an error to occur (because the object does not exist), and will not proceed correctly if the error does not occur. Thus, the virus cannot run from a web page. This might also serve as an anti-emulation trick in some environments.

If an error occurs, the virus uses the second parameter function to split a long string into its individual characters. The virus iterates through the characters in the resulting string, assigning one character to each of 12 variables until the entire string is decoded. Instead of using an 'if(condition)<body>', the virus uses a feature of JavaScript that is relatively little-known, but which is used very heavily in the js1kb demo world, where a Boolean evaluation that returns false will short-circuit the rest of the line in the case of an 'and' combination, and the true case will do the same for the 'or' combination. So, for example, instead of the following (which will perform the addition and assignment only if the length of wmfypxgq is not equal to eight):

```
if(wmfypxgq.length!=8)
        wmfypxgq+=wmfydagq,
        wmfydagq=wmfykngq()
```

the virus uses this:

```
wmfypxgq.length!=8&&(wmfypxgq+=wmfydagq,wmfydagq=wmf
ykngq())
```

JavaScript will evaluate the left half ('wmfypxgq. length!=8') and while the condition is met (that is, while the length is not equal to eight), it will execute the code in the right half (append the current character and fetch the next one). It should also be noted that the use of the comma allows the virus to omit the braces that would normally surround a multi-line body. This use of commas appears fairly consistently throughout the virus code. The use of the conditional shortcut, on the other hand, is highly erratic. This might suggest that multiple authors were involved, or perhaps just one author displaying different stages of development of the code.

The decoded strings are 'toString', 'charAt', 'charCodeAt', 'sort', a fake decryption key for the second text (see below), 'constructor', a base64-encoded encrypted string, 'fromCharCode', a base64 dictionary, a real decryption key, 'apply' and 'random'.

RDA, SCRIPT STYLE

After decoding the strings, the virus invokes the second of the anonymous functions in the array (the one that begins 'function(wmfydvgq,...'). This function generates up to 1,221 unique base-20 values to use as part of a decryption key for the decoded base64 strings. Each unsuccessful value is placed in an array so that it will not be used again. In the event that the key is not recovered after 1,221 attempts, the virus exits silently.

After each attempt at decrypting the text, the virus tries to run the resulting code. Instead of using the 'eval' function, or just declaring the code as a function and running it, the virus uses the 'array.sort.constructor' trick. This trick is derived from a way of obtaining a function reference by using only the alphabetic characters that can be generated using the minimum number of symbols (see the description of JJEncode [1] for the details). It has no special use in this context, since the virus has access to all possible characters. It is included simply to obfuscate the code further.

If the text has been decrypted correctly, the virus attempts to access the 'document' object. The access is performed inside a protected block, so that the virus can intercept any error that occurs. Once again, the virus is expecting an error to occur (because the object does not exist), and will not proceed correctly if that does not happen. This might also serve as an anti-emulation trick in some environments.

If an error occurs, the virus attempts to access the 'WScript' object. This access is also performed inside a protected block, so that the virus can intercept any error that occurs. However, in this case, the virus is not expecting an error to occur, and will not proceed correctly if one does. Specifically, if an error occurs, the virus fails to assign the real decryption key for the second text.

If the second text is decrypted correctly, the result is a block of code that is packed by Dean Edwards' JavaScript packer. This packer has remained enormously popular since its release in 2005, the 2007 release in particular – despite being outperformed by later packers such as JSCrush.

In any case, after unpacking and 'beautifying', we are left with a script of over 1,650 lines of dense code. There are no comments or blank lines. The code is a collection of 68 anonymous functions, some of which accept yet more anonymous functions as parameters, and some of which are not even used, such as the function to extract data from cookie files. There is no reason for such a large number of functions, other than to make the analysis more difficult.

The virus uses RC4 to decrypt an enormous array of strings, many of which are small enough to have been used as constants within the virus body, but again, they serve to make the analysis more difficult. The virus then attempts

to instantiate several objects: 'WScript.Shell', 'ADODB.Stream', 'Scripting.FileSystemObject', 'shell.application' and 'MSXML2.ServerXMLHTTP.6.0', and exits if any of them cannot be loaded.

ANTI-VM

The virus constructs nine arrays containing different groups of strings:

- One contains process names that the virus will attempt to terminate.
- One contains a word list that could be used as a dictionary attack, but which is not used by the virus.
- One contains a set of host names that the virus contacts in order to send and receive information.
- One contains a list of registry values relating to security policies.
- One contains a list of registry values relating to the *Windows* Security Center.
- One contains a list of domain suffixes which are used during URL generation.
- One contains a list of registry values relating to the *Windows Firewall* and the use of proxy servers.
- One contains a list of registry values relating to SafeBoot.
- One contains a list of registry values relating to the display of hidden files.

The virus uses the *Windows* Management Instrumentation interface to query the system configuration, as a virtual machine detection technique. The virus looks for a SCSI controller whose manufacturer name contains either 'Xen' (which appears twice in the list, perhaps in a copy-and-paste error, which suggests that the intended target is missing), 'Citrix', or 'Red Hat'; a BIOS whose manufacturer name contains 'innotek', 'Bochs', 'Xen', or 'QEMU'; a disk drive whose model name contains 'Bochs', 'VBOX', 'QEMU', 'Red Hat', 'VMware', 'Virtual HDD' (this is a typographical error – *VirtualPC*'s hard disk is named 'Virtual HD', and so the virus runs freely in *VirtualPC*), or 'Xen'; a process named 'CaptureClient.exe' (part of the *Capture* honeypot project); a computer system whose manufacturer name contains 'Parallels'; a processor whose manufacturer name contains 'Bochs' or 'QEMU'; or a computer name that contains either 'mcafee' or 'cnc-lab'. The matching of the computer name is case-insensitive. The virus exits if any of these is found.

FEELING INSECURE

The virus alters the registry to enable the hiding of files that have the hidden or system file attribute set. This is achieved by setting the 'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Hidden' registry value to the number 2. The virus constructs a registry value and data for writing. The registry value begins with 'lm', followed by two to five hexadecimal digits which are constructed in a convoluted fashion. The virus applies RC4 to the computer name and the 'lm' string, converts the result to a string of hexadecimal digits, and extracts some of the digits from the result. Thus, the value looks random but is actually constant on the given machine. An example of the registry data format is: 'C:\Program Files\[2–5 hex digits, but using 'ml' instead of 'lm' as the RC4 parameter]\[2–5 hex digits, using 'lm' as the RC4 parameter].js'. The virus attempts to write to the registry, but fails to specify a root, so the value is not created. This is a bug in the virus code.

The virus creates the registry value 'HKCU\Software\Microsoft\Windows\CurrentVersion\Run\[2–5hex digits, using 'cu' as the RC4 parameter]'. The data is set to the Application Data directory, for example, 'C:\Documents and Settings\me\Application Data\[2–5 hex digits, using 'uc' as the RC4 parameter]\[2–5 hex digits, using 'cu' as the RC4 parameter].js'.

The virus attempts to make many other changes to the registry – some of which are successful and some of which fail. It disables the *Windows* Security Center notifications by deleting the WSC registry value. It attempts to disable SafeBoot by deleting the registry key, but there is a bug in this code, and the attempt fails. It attempts to delete the SafeBoot registry key from HKCU, even though there is no 'System' hive in that location. It does, however, disable the *Windows Firewall* and the use of proxy servers, by changing their options in the registry. This is achieved by setting the 'HKLM\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\EnableFirewall' registry value, the 'ProxyEnable' and 'MigrateProxy' registry values under the 'HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings' registry key, and the 'HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\ParseAutoexec' registry value to zero.

The virus disables the *Windows* Security Center service by changing its start option in the registry. This is achieved by setting the 'HKLM\SYSTEM\CurrentControlSet\Services\wscsvc\Start' registry value to the number 4. The virus disables notifications in the *Windows* Security Center from the anti-virus and firewall services, and enables overrides for them. The virus is aware of the changes in registry layout between *Windows XP*, *Windows Vista* and later. For *Windows XP* compatibility, the virus achieves the effect by setting the 'UpdatesDisableNotify', 'FirewallDisableNotify', 'AntiVirusOverride', 'FirewallOverride' and

'AntiVirusDisableNotify' registry values under the 'HKLM\SOFTWARE\Microsoft\Security Center' registry key to the number 1. For *Windows Vista* and later compatibility, the virus achieves the same effect by setting the 'AntiVirusDisableNotify', 'FirewallDisableNotify' and 'FirewallOverride' registry values under the 'Security Center\Svc' registry key to the number 1.

The virus disables access to the command-interpreter, registry tools such as regedit, Task Manager, and the 'Display' option in the *Windows* Control Panel. This last one seems curious until you see that its name is 'NoDispCPL'. It seems likely that the virus writer thought that it meant 'No Display Control PaneL'. The effect of all of these is achieved by setting the 'DisableCMD', 'NoDispCPL', 'DisableRegistryTools', and 'DisableTaskMgr' registry values under the 'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies' registry key to the number 1.

The virus disables access to the 'HomePage' setting in the *Internet Explorer* control panel. This is achieved by setting the 'SOFTWARE\Policies\Microsoft\Internet Explorer\Control Panel\HomePage' registry value under both the 'HKCU' and the 'HKLM' registry hives to the number 1. The virus disables infection reporting from *MSRT*. This is achieved by setting the 'HKLM\SOFTWARE\Policies\Microsoft\MRT\DontReportInfectionInformation' registry value to the number 1. The virus disables the System Restore configuration. This is achieved by setting the 'HKLM\SOFTWARE\Policies\Microsoft\Windows NT\SystemRestore\DisableConfig' registry value to the number 1.

The virus disables the *Windows* Control Panel, the 'Windows Update' option, and the 'Folder' option from within *Windows Explorer*. This is achieved by setting the 'NoControlPanel', 'NoWindowsUpdate' and 'NoFolderOptions' registry values under the 'HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer' registry key to the number 1.

The virus enables the hiding of known file extensions in *Windows Explorer*. This is achieved by setting the 'HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\HideFileExt' registry value to the number 1. The virus disables System Restore. This is achieved by setting the 'HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SystemRestoreDisableSR' registry value to the number 1.

HOT PROSPECTS

The virus checks whether the directory 'C:\[2–5 hex digits, using 'prospect' as the RC4 parameter]' exists, and creates it if it does not. The virus sets the hidden and system file attributes on the directory in any case, and remembers if the directory was newly created. This state is checked later, and is used to decide whether the visible payload will execute. The virus attempts to open a file in that directory, whose name is '[2–5 hex digits, using 'it' as the RC4 parameter]'. If the file can be opened, then the virus reads it entirely. Otherwise, the virus creates the file, and writes to it the number of seconds since midnight on 1 January 1970. This could be considered the 'install time'.

The virus attempts to open a file whose name is '[2–5 hex digits, using 'r' as the RC4 parameter]'. If the file can be opened, then the virus reads it entirely. Otherwise, the virus opens its own file, reads it entirely, and then searches for what happens to be the last line in the virus code. This line is a long sequence of hexadecimal digits. The virus extracts 24 characters from the middle of the line, and uses it as a key to decrypt another string. The virus creates the originally requested file, and then writes the decrypted string to it. This might be a 'revision' number.

The virus attempts to open a file whose name is '[2–5 hex digits, using 'id' as the RC4 parameter]'. If the file can be opened, then the virus reads it entirely. Otherwise, the virus creates the file, and then writes a string of 12 random hexadecimal digits to it, converting to upper case if necessary. This is a machine-specific 'ID' that is used to communicate with the command-and-control server.

The virus attempts to open a file whose name is '[2–5 hex digits, using 'v' as the RC4 parameter]'. If the file can be opened, then the virus reads it entirely. Otherwise, the virus creates the file, and writes the virus filename to it.

START ME UP

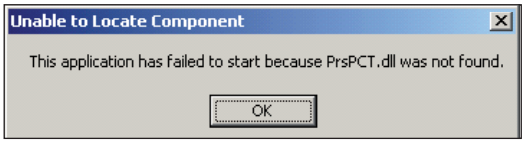
The virus enumerates files in the 'startup' directory for all users. The virus is aware of the different locations of that directory between the different versions of *Windows*. On *Windows XP* and earlier, it is '%userprofile%\Start Menu\Programs\Startup\'. On *Windows Vista* and later, it is '%userprofile%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\'. The virus deletes any '.js' files that exist in the directory, apart from any file whose name matches '[2–5 hex digits, using the current hour of the day as the RC4 parameter].js'. The virus opens its own file and reads up to the last line, calculates the new key to place in the last line, and then writes the combination to the 'startup' directory.

As a side note, the path of the startup directory is constant, no matter which locale is active, despite appearances to the contrary. Specifically, when viewing the directory in *Windows Explorer*, the name is localized so that, for example, the French version of *Windows* will show 'Menu Démarrer'. This should be obvious to a programmer,

given that there is no API to retrieve the path to the startup directory.

The virus creates the '%appdata%\[2–5 hex digits, using 'uc' as the RC4 parameter]' and '%programfiles%\[2–5 hex digits, using 'ml' as the RC4 parameter]' directories, as referenced above, and then hides them. It creates the '[2–5 hex digits, using 'lm' as the RC4 parameter].js' file in the Program Files hidden subdirectory, and the '[2–5 hex digits, using 'cu' as the RC4 parameter].js' file in the Application Data hidden subdirectory.

If the 'C:\[2–5 hex digits, using 'prospect' as the RC4 parameter]' directory was newly created, then the virus copies itself to '%temp%\[12 random hexadecimal digits].js', runs that copy, and then displays the following message:



The message will remain on the screen for 30 seconds, and then the original copy of the virus will exit, leaving the one in the temporary directory still running. Otherwise, the virus waits for a random amount of time, from slightly less than one second up to almost ten seconds, before continuing with the execution.

If the 'C:\[2–5 hex digits, using 'prospect' as the RC4 parameter]\[2–5 hex digits, using 'lock' as the RC4 parameter]' file exists, then the virus opens the file, reads it entirely, and checks that it contains only numbers. This file contains the date and time of the most recent execution of the code. The virus exits if the last execution was less than 15 seconds ago, since this is an indication that another copy is actively running. If the file does not exist, then the virus creates the file and writes the current time to it.

LOCK, STOCK, BARREL

The virus enumerates the CPUs and creates an array of the CPU names and number of cores. It also enumerates the video cards and creates an array of the video card descriptions.

The virus randomly reorders its list of hostnames, and then begins to enumerate the entries in the list. For each of the hostnames (currently: 'copertps.com', 'specrtop.org' and 'etpsoprc.ru'), the virus attempts to contact the host, send it a specific base64-encoded RC4-encrypted string, and receive another string in return. If a string is returned, the virus decodes and then decrypts it. If the resulting string contains the word 'prospect', then the host is accepted and will be used for any further requests for the next hour. If no string is returned or it does not decode correctly, then the virus continues the enumeration. If no acceptable host is found, then the virus generates a collection of URLs algorithmically, orders them randomly, and then attempts to contact each of the first ten in turn. For each of the algorithmic hostnames, the virus attempts to contact it and send it the specific string, as described above. If the proper string is returned, then that host will be used for the next hour.

The algorithm for URL generation is as follows: for each of the domain suffixes ('ru', 'net', 'info', 'in', 'eu', 'org', 'com', 'se', 'biz' and 'name'), the virus constructs a string in the format: 'prospect'.<month>.<date>.<four-digit year>.<domain suffix>. The virus hashes this string using a simple home-made algorithm, and then creates a new string of six to 12 lower-case letters, followed by the domain suffix. The virus constructs up to 10 unique URLs per domain, resulting in an array of potentially 100 entries (the count will be fewer if the hashes of any two of the URLs are identical).

If the host has a directory named 'u', then the virus fetches an update to its code from that location, and replaces the file containing the running script. However, the virus does not run this new file, so the update is not applied until later.

GETSYSTEMINFO()

Once per hour, the virus looks in the 'Application Data' and 'Appdata\Roaming' directories for each user, for the files named 'sitemanager.xml' and 'recentservers.xml' in a directory named 'FileZilla'. The virus reads either (or both if present) file in its entirety, and extracts some interesting properties from the files: host name, port, communication protocol, user name, and password. This information is uploaded to the 'r' directory on one of the hosts or generated URLs, as described above.

Once every 30 minutes, the virus calls a routine which now simply returns. However, enough of the code remains to determine that it would have uploaded files that were downloaded from a *WordPress*-hosted website. It would also have uploaded audio, video, graphical, and archive-format files that were requested from websites such as *Pinterest*, *Twitter* and *Sourceforge*.

The virus will, however, upload the complete system information to the 'k' directory on one of the hosts or generated URLs, as described above, and possibly receive a response containing commands to run. The information that is sent is:

- the uptime as measured in approximately 30-minute intervals
- a magic number that might identify the exact version of the virus

- the list of CPU names (see above)
- the computer name
- the number of CPUs
- the list of video card descriptions (see above)
- a value corresponding to the anti-virus software that is installed (see below)
- the account name for the logged-on user
- the current time zone
- a copy of the virus body
- a 'random' value (system-specific, as described above)
- the country code (see below)
- the *Windows* version
- the execution state of a particular process
- a virus-generated ID (see above)
- the processor architecture (32-bit or 64-bit)
- the language code (see below)
- the local time
- the special code that is appended to the virus body.

The virus determines which anti-virus software is installed by checking for the existence of the following directory names in the Program Files directory, and assigns each one a unique value:

Kaspersky Lab	Sophos	F-Secure
Spyware Doctor	Webroot	Avira
Panda Security	McAfee	ESET
Microsoft Security Essentials	Bitdefender	Sunbelt
Alwil Software	Symantec	COMODO
Microsoft Security Client	Trend Micro	AVG
AVAST Software	DrWeb	
Malwarebytes' Anti-Malware		

The virus uses the *Google* Geolocation services to determine the country code for the host IP address. The API in question has been deprecated since 2010, but continues to be available for a limited number of requests. In the case of the virus, it needs to make only one request.

The particular process that interests the virus is an .exe file with a name which the virus generates by using the key 'btcm'. If the process is found to be running, then the virus sets the priority to run only when the system is idle.

The language code is determined by requesting the language version of the operating system, and then looking up the corresponding entry in the RFC1766 MIME database.

COMMAND AND CONQUER

The virus can receive a list of commands to execute. The commands are very short: 'e', 'hp', 'r', 'd', 'fbc', 'dbs', 'b', 'u', 'fbl', 'redu' and 'fbf'.

The 'e' command can be used to run arbitrary script code where the results are not checked.

The 'hp' command is intended to be used to redirect all URL connections to the requested site. This would be achieved by placing the site name in the appropriate protocol under the 'Prefixes', 'DefaultPrefix' and 'Prefixes\www' registry keys under the 'HKLM\Software\Microsoft\Windows\CurrentVersion\URL' registry key. However, there is a bug in this code, which means that the command does not work.

The 'hp' command can set the Start Page in *Microsoft Internet Explorer*. This is achieved by setting the 'Software\Microsoft\Internet Explorer\Main\Start Page' registry value in both the 'HKCU' and the 'HKLM' hives. The command can optionally change the start page in *Google Chrome*. This is achieved by changing the appropriate settings in the '%userprofile%\Local Settings\Application Data\Google\Chrome\User Data\Default\Preferences' file. If the *Chrome* option is selected, then *Mozilla* will be targeted, too. The virus searches for the 'user.js' file in the subdirectories of the '%appdata%\Mozilla\Firefox\Profiles' directory. If the file is found, then the virus will change the start page in that file.

The 'r' command can be used to run any executable files on the local system.

The 'd' command can be used to download and run a specified file from a specified URL. The virus will contact the server and wait up to approximately seven seconds for a response.

The 'fbc' command was probably a routine used to start a chat on *Facebook*, but the code is not present in this version of the virus.

The 'dns' command can be used to change the DNS server on the local system. This is achieved by changing the 'DhcpNameServer' and 'NameServer' registry values under the 'HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters' registry key.

The 'b' command can be used to download an .exe file whose local name the virus generates by using the key 'btcm'. The virus will download the file only if the parameters for its execution are different from the previous execution, if any. If the download is requested, then the virus stops the existing 'btcm' process, if it is running, and then downloads and runs the new file. The virus intends to return the state of execution of the new file, but there is a bug in this code so the execution state always appears to be a failure.

The 'u' command can be used to update the virus code dynamically. If the virus has been updated successfully, then it clears the file that holds the last execution time, in order to allow the new code to start running without failing the '15 seconds' check.

The 'fbl' command was probably a routine used to 'Like' a page on *Facebook*, but the code is not present in this version of the virus.

The 'redu' command can be used to run arbitrary script code that accepts a single parameter, for example solving equations. The results will be uploaded to the 'reduce' directory on one of the hosts or generated URLs, as described above.

The 'fbf' command was probably a routine used to become a fan of a *Facebook* page, or to send a 'friend' request, but the code is not present in this version of the virus.

After all commands have been processed, and if the 'b' command has not been received, the virus stops the 'btcm' process, if it is running. It is unknown what this process does.

The virus periodically spends five seconds alternating between sleeping for one second and enumerating the list of running processes. The virus attempts to terminate any process whose name contains any of the following strings:

rubotted	avg	avast	autoruns
tcpview	msconfig	hijack	otl
fs20	msss	filemon	minitool
systemlook	mrt	jrt	wireshark
unlocker	procmon	mse	sdasetup
mbam	clean	rkill	ccsetup
resmon	procexp	fss	rstrui
housecall	ptinstall	npe	wuauclt
mcshield	sdefendi	regmon	issetup
mbsa	fiddler	zoek	gmer
roguekiller	dds	emergencykit	exeradar
avenger	hitman	combofix	perfmon
reged	spybot	klwk	eset
windows-kb	hotfix		

In all cases except for the 'hotfix' entry, the matching is case-insensitive. The case-sensitivity of the hotfix entry appears to be a bug in the virus code.

AUTORUN.INFECT

The virus repeatedly enumerates the list of drives, waiting for a USB device to be inserted. When a USB device is found, the virus creates a directory on each of its drives, '\[2–5 hex digits, using 'usb' as the RC4 parameter]', and then hides this directory. The virus places a file inside the directory, '\i[2–5 hex digits, using 'lnk' as the RC4 parameter].js'. For every other directory in the root of the drive, excluding any named 'recycled', the virus creates a shortcut using the name of the directory followed by '.lnk'. The virus then hides the original directory. The icon for the shortcut is the folder icon, but the shortcut arrow is added to the corner of it (there is a registry change that can make it go away, but the virus does not make use of it). In any case, the action of the shortcut is to run the virus script, and then open an *Explorer* window showing the contents of the directory.

The virus places another file inside the hidden directory, '\g[2–5 hex digits, using 'ar' as the RC4 parameter].js', and then creates an 'autorun.inf' in the root directory of each of the drives on the USB device. The virus writes a random number of lines (from 35 to 100) of random text. For each of those lines, there is a 20% chance that the virus creates a section with a random name. The name is a random number from five to 10 characters. Otherwise, the line is an assignment using a random number from 10 to 30 characters on each side of the equals sign. Then the virus alternates between writing five to 10 random lines and one real line. The order of the real lines ('shell\explore\command=', 'shell\open\command=', 'open=' and 'shellexecute=') is also random. After all of the real lines have been written, the virus writes another random number of lines (from 15 to 50) of random text. For each of those lines, there is a 20% chance that the virus creates a section with a random name. Otherwise the line is an assignment, as before.

CONCLUSION

One of the main problems with describing code that can update itself is that no two descriptions will be alike. The code could update itself in different ways, depending on certain circumstances – for example, different countries might be served different versions. Even requests at different times of the day might yield different results. The best that we can say is that 'this sample, with this hash value, behaves in this way' – and that's not saying much. Fortunately, the different variants that we have seen have a similar overall structure, which allows us to detect them generically. That's all we need to say.

REFERENCES

[1] Ferrie, P. $$$_+$$+$$__+_$+$$_$+$$$_+$$_$. Virus Bulletin, February 2011, p.4. http://www.virusbtn.com/pdf/magazine/2011/201102.pdf.

MALWARE ANALYSIS 3

NEDSYM SPAMMING

He Xu
Fortinet, Canada

A number of security reports in 2012 declared that spam was on the decline [1]. However, spam still accounts for more than 70% of all email sent – an enormous proportion. Why does this happen? In this article we will expose the tip of the iceberg by analysing a recent spambot which is driven by the Andromeda botnet and detected by *Microsoft* as Win32/Nedsym.G.

INSTALLATION

The bot uses a loader and mailer module mechanism.

The loader will create a new folder in the %App Data% system folder and generate an extremely long folder name using the following hard-coded string prefixed with 'x':

qwertyuiopasdfghjklzxcvbnm123456789

Then, if the full path of the executing bot does not include the string 'vcnost.e', it will enumerate all processes and terminate every one whose filename includes 'svcnost.' to make sure that only one instance of the bot is running.

It then moves itself to the sub-folder as svcnost.exe.

Next, the bot creates another folder with the same name as the previous one, but with the character '2' added to the end. We will see why later.

It adds the following registry value to ensure that it runs automatically each time the system is started:

Key: HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Name: Windows Init

Value: %App Data%\x<Random String>2\svcnost.exe

The bot moves itself again so that its location matches the registry value.

It starts itself in the new location and then overwrites the system hosts file, as shown in Figure 1, in order to disable various security software updates, including those from *Kaspersky Lab*, *McAfee*, *Symantec* and *Microsoft* – 195 in total.

We have found newer variants that simplify and customize this routine. First, the malware tries to create a named mutex, 'MSCTF.Shared.MUTEX.LDR', to prevent multiple instances running, then it copies itself to %App Data% with the hard-coded filename 'wmprwise.exe' and adds the following registry entry:

```
  1  127.0.0.1 downloads4.kaspersky-labs.com
  2  127.0.0.1 downloads3.kaspersky-labs.com
  3  127.0.0.1 downloads2.kaspersky-labs.com
  4  127.0.0.1 downloads1.kaspersky-labs.com
  5  127.0.0.1 downloads-us1.kaspersky-labs.com
  6  127.0.0.1 rads.mcafee.com
  7  127.0.0.1 www.secuser.com

 32  127.0.0.1 download1.avast.com
 33  127.0.0.1 upgrade.bitdefender.com
 34  127.0.0.1 windowsupdate.microsoft.com
 35  127.0.0.1 www.lavasoftusa.com

187  127.0.0.1 www.sophos.com
188  127.0.0.1 www.sophos.com
189  127.0.0.1 www.symantec.com
190  127.0.0.1 www.symantec.com
191  127.0.0.1 www.trendmicro.com
192  127.0.0.1 www.trendmicro.com
193  127.0.0.1 www.viruslist.com
194  127.0.0.1 www.viruslist.ru
195  127.0.0.1 www3.ca.com
```

Figure 1: Compromised hosts file.

Key: HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

Name: Microsoft Firewall 2.9

Value: %App Data%\wmprwise.exe

PREPARATION

Add to firewall list

The bot adds its path to the firewall's list of authorized applications by adding the following registry entry:

Key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\AuthorizedApplications\List

Name: %App Data%\x<Random String>2\svcnost.exe

Value: %App Data%\x<Random String>2\svcnost.exe:*:Enabled:ldrsoft

The embedded module will be decrypted and extracted in memory by simulating the PE loader behaviour, then the loader will call the entry point of the module to run the malicious code there. As we see in Figure 2, the original module's name is Mailer.dll, and it has export structure, but no export function.

Create thread

As shown in Figure 3, the module simply creates a single new thread then returns.

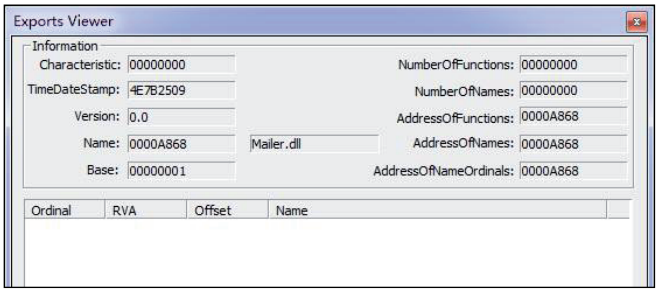


Figure 2: The mailer module.

```
DllEntryPoint   proc near               ; DATA XREF: HEADER:0

ThreadId        = dword ptr -4
hinstDLL        = dword ptr  8
fdwReason       = dword ptr  0Ch
lpReserved      = dword ptr  10h

                push    ebp
                mov     ebp, esp
                add     esp, 0FFFFFFFCh
                push    esi
                mov     eax, [ebp+fdwReason]
                dec     eax
                test    eax, eax
                jnz     short loc_10009526
                lea     eax, [ebp+ThreadId]
                push    eax             ; lpThreadId
                push    0               ; dwCreationFlags
                push    0               ; lpParameter
                push    offset Thread01 ; lpStartAddress
                push    0               ; dwStackSize
                push    0               ; lpThreadAttributes
                call    j_CreateThread

loc_10009526:                           ; CODE XREF: DllEntry
                pop     esi
                xor     eax, eax
                inc     eax
                leave
                retn    0Ch
DllEntryPoint   endp
```

Figure 3: The module entry point only creates one thread.

```
                mov     eax, ds:pMZ
                call    Ldr_DLL_Run

LOOP_01:
                push    1388h
                call    j_Sleep
                jmp     short LOOP_01
; ---------------------------------------
                retn
```

Figure 4: Loader drops in a dead loop for sleeping.

The loader will then drop in a dead loop that sleeps permanently (Figure 4).

The new thread performs the same operation as the new variant – it attempts to create a mutex named 'LDR.ML.STARTED' and updates the following registry entry (the value string should be the current BotID, which is generated randomly):

Key: HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\LowRegistry

Name: SavedLegacySettingsML

Value: 447140859

Drop and load two additional modules

To encrypt/decrypt packages and support mail server connection by SSL, the bot drops and decrypts two additional DLLs:

DLL drop path and filename	Original filename in export table
%AppData%\desktop.ini	BTREE.dll
%AppData%\ntuser.dat	zlib1.dll

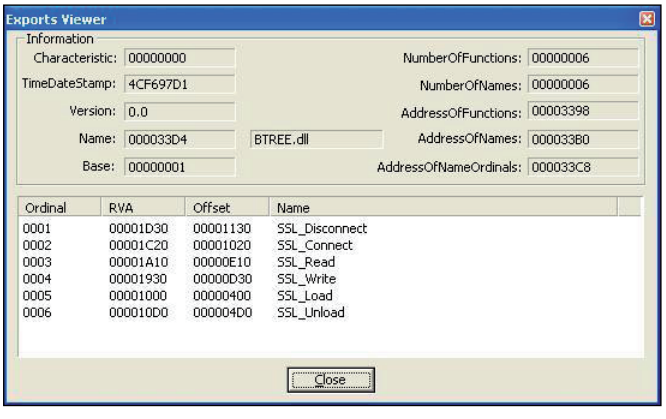


Figure 5: The dropped DLL desktop.ini export table.

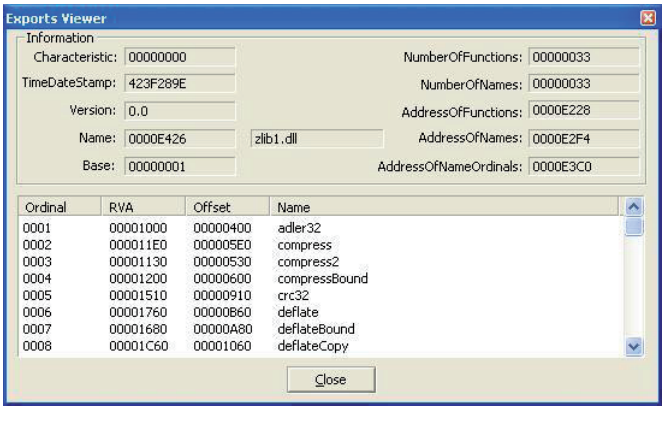


Figure 6: The dropped DLL ntuser.dat export table.

All the export functions will be loaded into the bot and will be used for SSL connection with the mail server.

The bot only needs a handful of APIs from this module for encrypt/decrypt packages (Figure 7).

```
push    38h
push    offset a1_2_3   ; "1.2.3"
push    2Fh
push    ebx
push    inflateInit2_
pop     eax
call    eax
push    4
push    ebx
push    inflate
pop     eax
call    eax
test    eax, eax
jz      short LOOP_01   ;
push    ebx
push    inflateEnd
pop     eax
call    eax
```

```
push    38h
push    offset a1_2_3   ; "1.2.3"
push    0
push    8
push    0Fh
push    8
push    -1
push    ebx
call    deflateInit2_
push    4
push    ebx
push    deflate
pop     eax
call    eax
test    eax, eax
jz      short LOOP_01   ;
push    ebx
push    deflateEnd
pop     eax
call    eax
```

Figure 7: The imported APIs from DLL ntuser.dat.

COMMUNICATION

Different gates for different jobs

Most strings are encrypted initially, so before talking with the C&C server, the following gates are extracted:

/stat1.php	/u.php
/stat2.php	/error.php
/smtps.php	/logacc.php

Fake traffic

Similar to the Pushdo botnet, the bot will first send fake traffic to make monitoring more difficult. It will select one IP address from the following IP ranges:

```
"89.149.242.%RND_NUMBER[1-254]"|
"89.149.243.%RND_NUMBER[1-254]"|
"89.149.244.%RND_NUMBER[1-254]"|
"217.20.115.%RND_NUMBER[1-254]"|
"217.20.127.%RND_NUMBER[1-254]"|
"217.20.112.%RND_NUMBER[1-254]"
```

The initial IP ranges.

```
<?"89.149.242.149"|"89.149.243.167"|"89.149.244.62"|"
217.20.115.110"|"217.20.127.37"|"217.20.112.235"?>
```

The IP ranges after interpretation.

Our investigation revealed that all the IP ranges belong to a single organization (see Figure 8).

```
inetnum:        89.149.241.0 - 89.149.244.255
netname:        NETDIRECT-NET
descr:          Leaseweb Germany GmbH (previously netdirekt e. K.)
```

Figure 8: The IP ranges belong to a single organization.

The bot uses the following string as the fake package, and encrypts the package using the deflate API from the dropped ntuser.dat module:

```
ver=1&login=kuklachev&macroses_version=2&SMTPWo
rking=True&SMTPOn=True&SMTPSentNumber=5.x
```

As shown in Figure 9, there is no response to the fake traffic.

Stream Content

```
POST /stat1.php HTTP/1.0
Host: 89.149.243.167
User-Agent: Mozilla/4.0 (compatible. MSIE 8.0. Windows
NT 5.1)
Accept-Encoding: gzip,deflate
Content-Length: 81

x.+K-.5T..O.....IL.H-S.ML../N-./K-*....5R..
.../...K.
)*M....!...y%~..I@.M.kJ.X
```

Figure 9: Fake traffic example.

Get initial configuration

For the real traffic, the following pattern is used to generate the package:

```
ver=%s&login=%s&macroses_version=2&SMTPWorkin
g=%s&SMTPOn=%s&SMTPOffMessage=%s&SMTPB
lockTime=%d&SMTPSentNumber=%d&botid=%s&las
tsmtp=%s
```

The interpreted parameters are as follows:

```
ver=200&login=admin6&macroses_version=2&SMTPW
orking=untested&SMTPOn=True&SMTPOffMessage=
&SMTPBlockTime=0&SMTPSentNumber=0&botid=44
7140859&lastsmtp=
```

After deflation, the package will be sent to the C&C server with a POST header. There is a trick in the HTTP POST header: the host is not the real C&C address but a hard-coded IP. The gate file on the C&C server is http://sonymaind20k.ru/stat1.php.

After inflation we see the configuration file shown in Figure 11.

To check the validity of the received package, the bot will examine the '#BODY' tag. The first line of configuration data is the list of victim email addresses and will be used for the SMTP 'RCPT TO:' command [2]. The second line is the email address for the SMTP 'MAIL FROM:' command. The third line is the spamming job ID that is issued by the C&C server, including username and password combined with the '/' symbol. The fourth line is the proxy server. The fifth line is the mail server port mapping table. From the sixth line to the first #BODY tag is the block for updating the bot's initial options (see Figure 12).

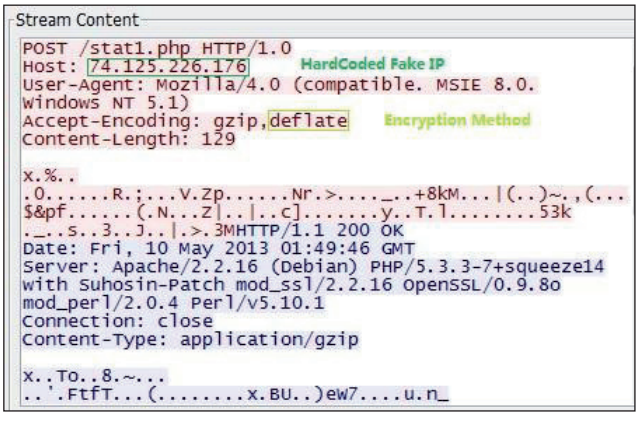


Figure 10: First real traffic.

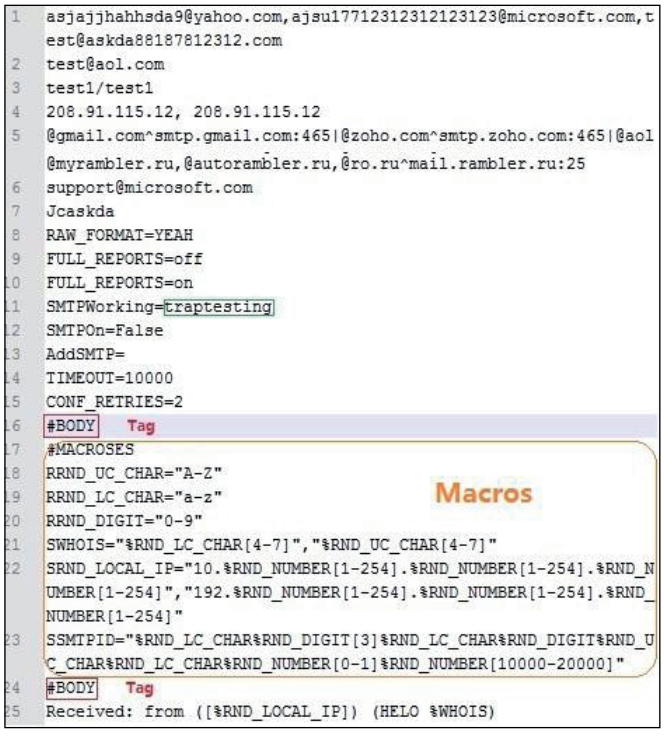


Figure 11: Decrypted configuration.

```
mov     Max_Threads, 1
mov     TimeOut_2710h, 2710h
mov     CONF_RETRIES, 1
mov     MAXSMTPFails, 5
mov     DD_SMTPBlockTime, 0
mov     DD_SMTPSentNumber, 0
mov     OneTimeSMTPid, 0
mov     Tag_SMTPOn, 0
mov     Tag_FULL_REPORTS, 0 ;
```

Figure 12: Initial options.

The bot can support the following keyword options, and skips lines that it cannot support:

RAW_FORMAT

CONF_TIMEOUT

CONF_RETRIES

MAXIMUM_THREADS

MAXSMTPFails

OneTimeSMTPid

FULL_REPORTS

OneTimeSMTP

SMTPOn

Remarks

In the example shown in Figure 11, the C&C server /stat1.php will update bot RAW_FORMAT to YEAH, then turn off and turn on the FULL_REPORTS option. So finally the C&C server has only updated the bot's CONF_RETRIES value from the default 1 to 2.

In fact, we can see that the SMTPWorking value is 'traptesting', which means that it is not really for spamming but for grabbing local email information in the next section.

Define macro

The block between the first and second #BODY tags is the macro pattern and range limit.

There are two different types of macro. The first type, starting with the character 'R', is for declaring a random range; the second type, starting with 'S', is for declaring the format. The bot will convert each macro beginning with 'S' to a fixed string that may be used for spamming later.

Grab local emails

The bot will try to grab local email information from the following applications:

- *RITLabs The Bat! Email Client*
- *Internet Explorer IntelliForms*
- *Mozilla Firefox*

Once any information is found and grabbed, the bot will convert it to an extremely long string and add it into the following pattern:

botid=%s&accs=%s

Then it deflates the information and sends it to the C&C server http://sonymaind20k.ru/logacc.php with a similar HTTP POST header to the one shown in Figure 10.

For the grabbed emails, the bot will test the data directly. It will generate an email template according to the pattern that follows the second #BODY tag in the configuration file.

The bot begins to enter a permanent loop for getting the spam configuration file and spamming routine.

Get spamming template

The bot will send the same package to the C&C server as it sent to get the initial configuration, but this time the C&C server gate path will be changed to http://sonymaind20k.ru/stat2.php. In response, it gets the spamming configuration file, as shown in Figure 13.

This set of configuration data is similar to the first. The first line is the list of victim email addresses separated by the symbol ‘,’. The second line is a macro to indicate that the bot should get the email from the LAST_GOOD_MAIL block, or from the first line if the block is empty.

According to the configuration, the bot will run 30 threads to send spam (Figure 13).

More detail

In each spamming thread, the bot will get a single address from the list of victim email addresses to use as the destination, then pick up another from the LAST_GOOD_MAIL block to use as the ‘MAIL FROM:’ content, continually interpreting the final spamming email according to the template that follows the second #BODY tag.

The bot currently supports the following keywords in the spam template:

- RND_UC_CHAR
- RND_LC_CHAR
- RND_DIGIT
- RND_NUMBER
- CURRENT_DATE_TIME
- RND_DATE_TIME
- OUTLOOK_BOUNDARY
- OUTLOOK_MESSAGE-ID
- OUTLOOK_SHORT_MESSAGE-ID
- PROXY
- TO_MAIL
- LOCAL_HOST
- TO_NAME
- TO_CC_DEFAULT_HANDLER
- LAST_GOOD_MAIL
- FROM
- FROM_NAME

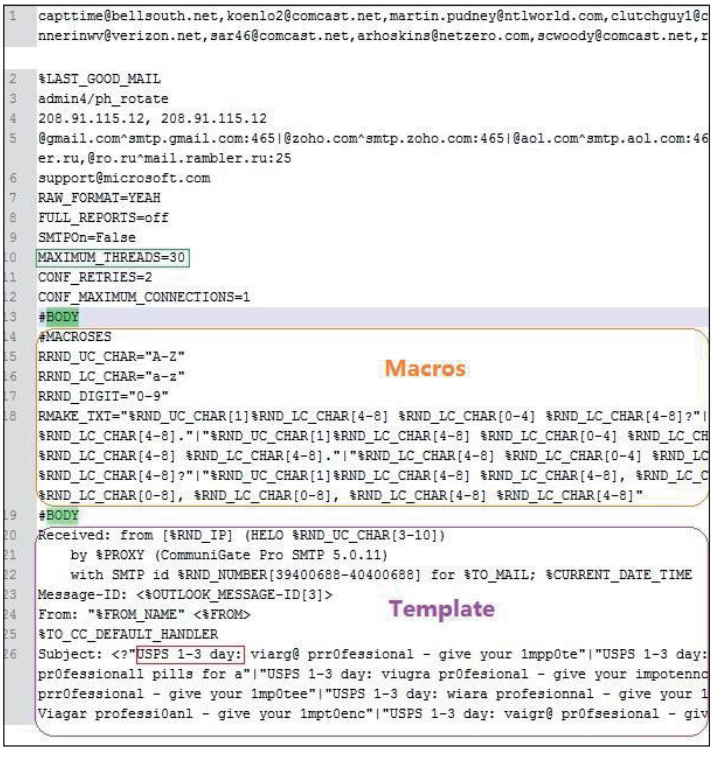


Figure 13: Second configuration.

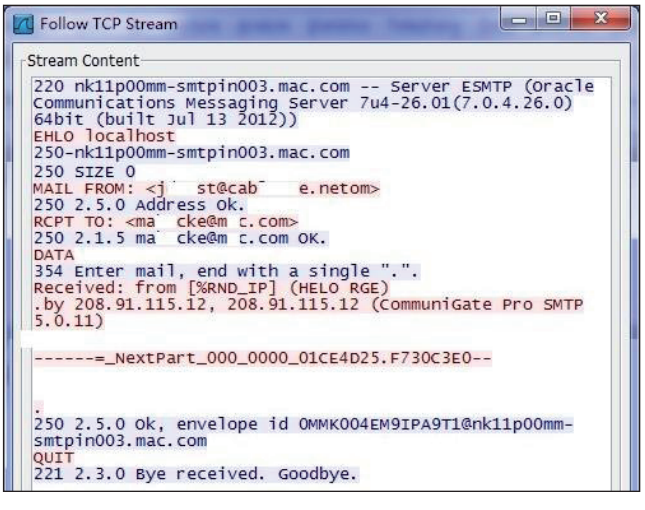


Figure 14: SMTP traffic.

After interpretation, all keywords will be converted to the final spam email content and sent to the mail server according to the SMTP commands (Figure 14).

An example of the spam email is shown in Figure 15.

The referred URL in the spam message is an online pharmacy – Figure 16 shows a screenshot of the site.

Figure 15: Example of the spam email.

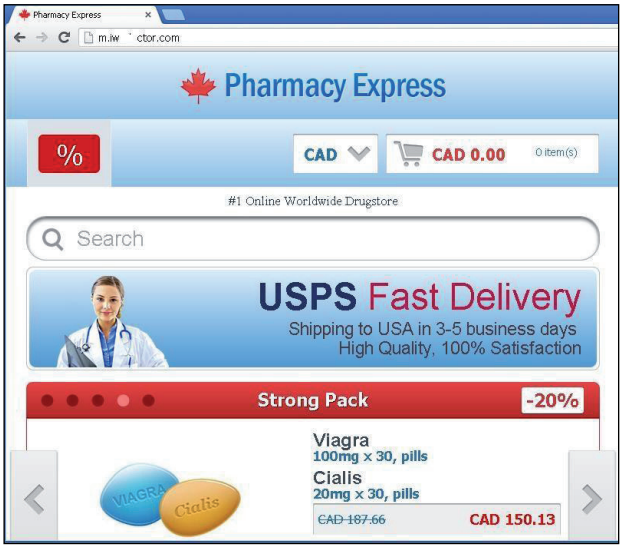


Figure 16: URL screenshot.

When the bot has tried all the addresses, it will send a feedback package to the C&C server using the following pattern:

```
ver=%s&login=%s&id=%s&totalsent=%d&totallost=%d
&totaldrop=%d&SMTPSent=%d&Remarks=%s&macros
es_version=2&botid=%s&lastsmtp=%s&SMTPWorking=
%s&sent=%s&lost=%s&drop=%s
```

The bot does not need the C&C server to respond, so once the feedback package has been sent, it will close the connection and prepare for the next spamming routine (Figure 17).

Error feedback in SEH routine

To improve the bot's next generation and fix possible bugs, it has added code for feedback traffic in SEH routines.

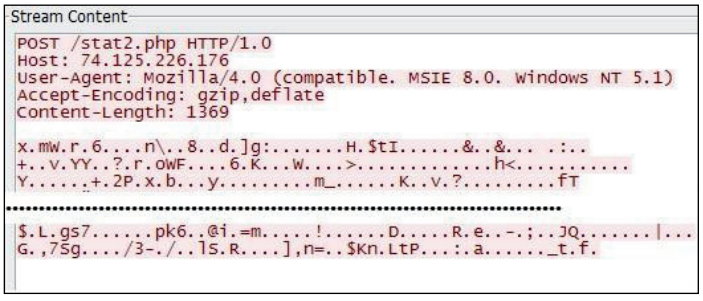


Figure 17: The feedback package is one-way only, so the bot closes the connection.

When an exception occurs, the bot will not try to fix it, but will generate a package using the following pattern and send it back to the /error.php C&C server:

```
id=%s&botid=%s&error=%d
```

The bot feeds back one of the following error codes to the C&C server:

Error code	Description
1	Keywords interpretation exception
2	Spam template interpretation exception
3	Convert string or number range exception
4	Pattern interpretation exception

CONCLUSION

Through our analysis of this spambot, we have seen a glimpse of how it spreads and works. The bot herder has never given up any opportunities to earn money. Each time we thought that the situation was improving, the bot herder was ready to launch a new round of attacks. It is a good idea to upgrade insecure email servers to ensure that they block spam.

REFERENCES

[1] Gudkova, D. Kaspersky Security Bulletin Spam Evolution 2012. January 2013. http://www.securelist.com/en/analysis/204792276/Kaspersky_Security_Bulletin_Spam_Evolution_2012.

[2] RFC 821 Simple Mail Transfer Protocol. http://tools.ietf.org/html/rfc821.

SPOTLIGHT

GREETZ FROM ACADEME: ON MOTIVATION

John Aycock
University of Calgary, Canada

Some academics are fascinated by what young men do with their computers. I don't mean that in a licentious, 'Quick! Censor the Internet!' way, though. It seems that every few years I stumble across an academic article written about the dreaded hackers, their motivations, and what can be done about them. Invariably this involves interviews with adolescents or young men – at least it does in the cases where the academics bother to track them down (some papers just survey university students and call it a day, but they shall remain citeless). In this context, the focus on individuals bearing the Y chromosome does seem to be largely accurate, statistically speaking, along with the age bias. (Perhaps there is a hidden demographic of senior citizens who hack, but presumably they're too wily and treacherous to be caught.)

My latest find was in a recent issue of the *Communications of the ACM*, or *CACM* for short. *CACM* is the premier publication of the Association for Computing Machinery, a decades-old organization to which many academics belong, giving them access to a massive digital library along with a really spiffy membership card (*VB*, take note!). Xu *et al.*'s paper 'Why Computer Talents Become Computer Hackers' [1] is yet another foray into the hacking arena, albeit with a Chinese cultural focus – both a strength and weakness that the authors note. The basis of their paper: the expected interviews with six young male hackers.

The academic focus on the portrait of the hacker as a young man may seem somewhat puzzling to those on the front lines of the anti-malware industry, and even those academic security researchers who do very applied work such as botnet infiltration. Having spent time in both camps, I think the anti-malware industry surpasses the academics with respect to colourful names – I remember one anti-malware conference where an attendee in the audience referred to malware writers as 'scum-sucking pigs'. And that was *before* the bar had opened.

In all seriousness, though, surely most security havoc is now caused by more seasoned professionals. Xu *et al.* suggest that there is a progression over time from 'affection for computers' to 'curious exploration' to 'illicit excursion' to 'criminal exploitation' [1]. As a cynic, I would interpret this progression to mean that curiosity and affection for computers should mercilessly be stamped out. The authors' conclusion from their research is somewhat different: 'Eliminating tolerance and strengthening moral-value constraint appear to be the only manageable options in resisting hacking today.'

Whenever I read academic papers on this topic, I'm reminded of Sarah Gordon's work on virus writers (e.g. [2, 3]), which regrettably is often overlooked in academic papers, even though she published a relatively recent commentary on it in an academic venue [4]. Again, the gap between industry and academia rears its head.

Although I callously spoiled Xu *et al.*'s paper by giving away its conclusion above, the periodic nature of academic 'hacker motivation' papers ensures that there are more. A particular favourite of mine was also published in *CACM*, in 2005. McHugh and Deek argued that a sandboxed 'microcosm' in which hackers could unleash their malware safely would be good for the Internet as a whole [5]. They delved into hacker motivation to address the rhetorical question: 'Is the system we propose likely to attract the interest of hackers?' Somehow I think that people in the anti-malware industry would have a quick, realistic answer to that.

With all this discussion of young hackers, I should disclose my own age bias. I'm old enough to remember the use of the word 'hacker' in a positive sense [6]. Each time I write the word 'hacker' in the popular, derogatory way, I involuntarily grit my teeth and twitch slightly. For those who can't appreciate this particular quirk of mine, simply imagine the word 'virus' being used to describe all instances of malware as well as Grandma's hard drive in need of a defrag.

REFERENCES

[1] Xu, Z.; Hu, Q.; Zhang, C. Why Computer Talents Become Computer Hackers. Communications of the ACM 56(4), April 2013, pp.64–74.

[2] Gordon, S. The Generic Virus Writer. Proceedings of the 4th Virus Bulletin International Conference, 1994.

[3] Gordon, S. The Generic Virus Writer II. Proceedings of the 6th Virus Bulletin International Conference, 1996.

[4] Gordon, S. Understanding the Adversary: Virus Writers and Beyond. IEEE Security & Privacy, September/October 2006, pp.67–70.

[5] McHugh, J. A. M.; Deek, F. P. An Incentive System for Reducing Malware Attacks. Communications of the ACM 48(6), June 2005, pp.94–99.

[6] Hacker. Jargon File, version 4.4.7. E. Raymond, ed., 2003. http://www.catb.org/jargon/html/H/hacker.html.

END NOTES & NEWS

Cyber Intelligence Europe takes place 17–19 September 2013 in Brussels, Belgium. For details see http://www.intelligence-sec.com/events/cyber-intelligence-europe.

Hacker Halted USA will take place 19–21 September 2013 in Atlanta, Georgia, USA. For more information see https://www.hackerhalted.com/2013/us/.

The (ISC)2 Security Congress 2013 takes place 24–27 September in Chicago, IL, USA. For details see https://congress.isc2.org/.

InfoSecurity Russia will be held 25–27 September in Moscow, Russia. For details see http://www.eng.infosecurityrussia.ru/.

VB2013 takes place 2–4 October 2013 in Berlin, Germany. The conference programme and online registration are now available. See http://www.virusbtn.com/conference/vb2013/.

SecTor 2013 takes place 7–9 October 2013 in Toronto, Canada. For details see http://www.sector.ca/.

Hactivity 2013 takes place 11–12 October 2013 in Budapest, Hungary. For details see https://hacktivity.com/en/.

ISSE 2013 will take place 22–23 October 2013 in Brussels, Belgium. For more details see http://www.isse.eu.com/.

MALWARE 2013 takes place 22–24 October 2013 in Fajardo, Puerto Rico, USA. See http://www.malwareconference.org/.

Ruxcon 2013 takes place 26–27 October 2013 in Melbourne, Australia. See http://www.ruxcon.org.au/.

RSA Conference Europe takes place 29–31 October 2013 in the Netherlands. For details see http://www.rsaconference.com/events/2013/europe/index.htm.

The First Workshop on Anti-malware Testing Research (WATeR 2013) takes place on 30 October 2013 in Montreal, Canada. For full details see http://secsi.polymtl.ca/water2013/.

Oil and Gas Cyber Security will be held 25–26 November 2013, in London, UK. For details see http://www.smi-online.co.uk/2013cyber-security5.asp.

AVAR 2013 will take place 4–6 December 2013 in Chennai, India. For details see http://www.aavar.org/avar2013/.

Botconf 2013, the 'first botnet fighting conference', takes place 5–6 December in Nantes, France. For details see https://www.botconf.eu/.

FloCon 2014 will be held 13–16 January 2014 in Charleston, SC, USA. For details see http://www.cert.org/flocon/.

RSA Conference 2014 will take place 24–28 February 2014 in San Francisco, CA, USA. For more information see http://www.rsaconference.com/events/us14.

VB2014 will take place 24–26 September 2014 in Seattle, WA, USA. More information will be available in due course at http://www.virusbtn.com/conference/vb2014/. For details of sponsorship opportunities and any other queries please contact conference@virusbtn.com.

SUBSCRIPTION RATES

Subscription price for Virus Bulletin magazine (including comparative reviews) for one year (12 issues):

- Single user: $175
- Corporate (turnover < $10 million): $500
- Corporate (turnover < $100 million): $1,000
- Corporate (turnover > $100 million): $2,000
- *Bona fide* charities and educational institutions: $175
- Public libraries and government organizations: $500

Corporate rates include a licence for intranet publication.

Subscription price for Virus Bulletin comparative reviews only for one year (6 VBSpam and 6 VB100 reviews):

- Comparative subscription: $100

See http://www.virusbtn.com/virusbulletin/subscriptions/ for subscription terms and conditions.

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire OX14 3YP, England

Tel: +44 (0)1235 555139 Fax: +44 (0)1865 543153

Email: editorial@virusbtn.com Web: http://www.virusbtn.com/