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

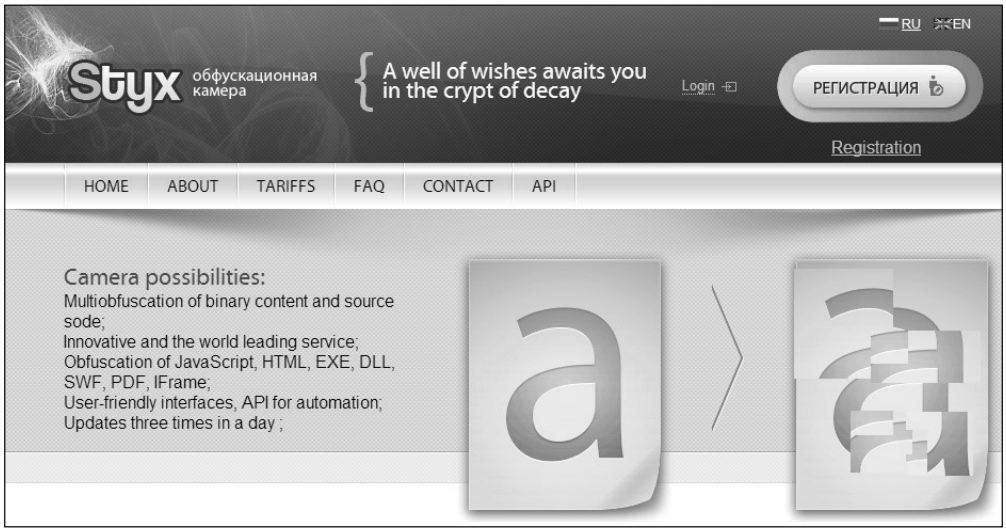

Figure 1: Original Styx service provider.

```
# 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.

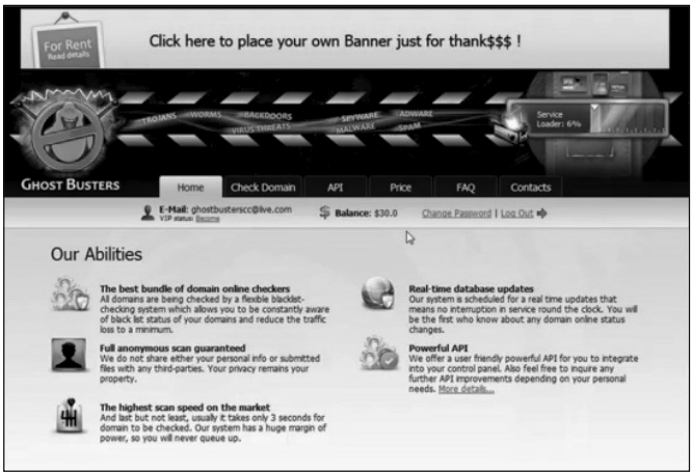

Figure 2: Ghost Busters service.

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.

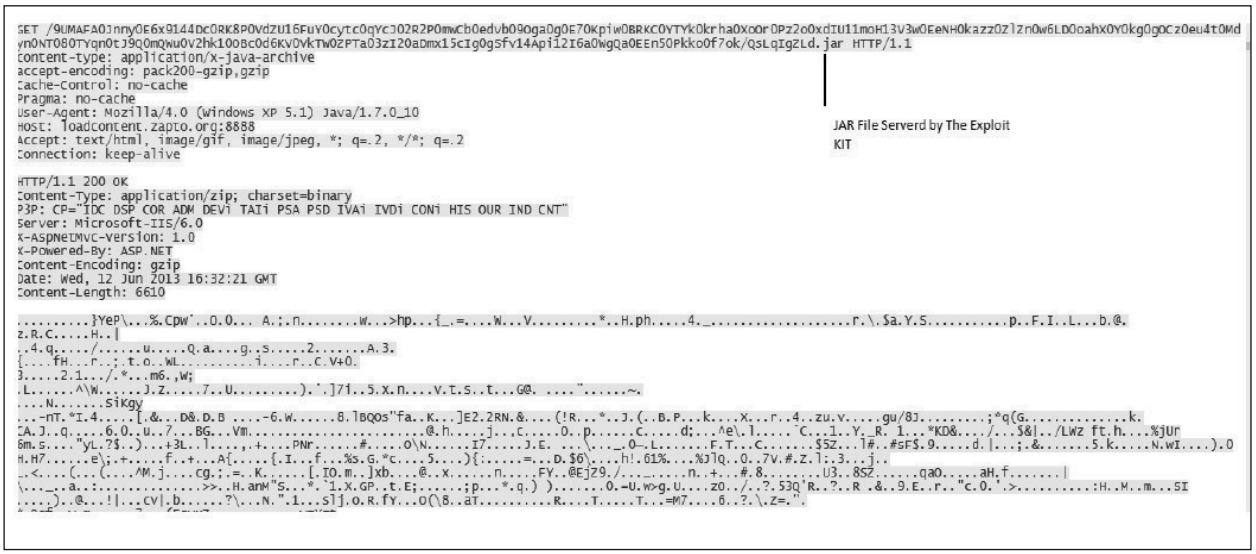

Figure 3: Malicious Jar file used by Styx.

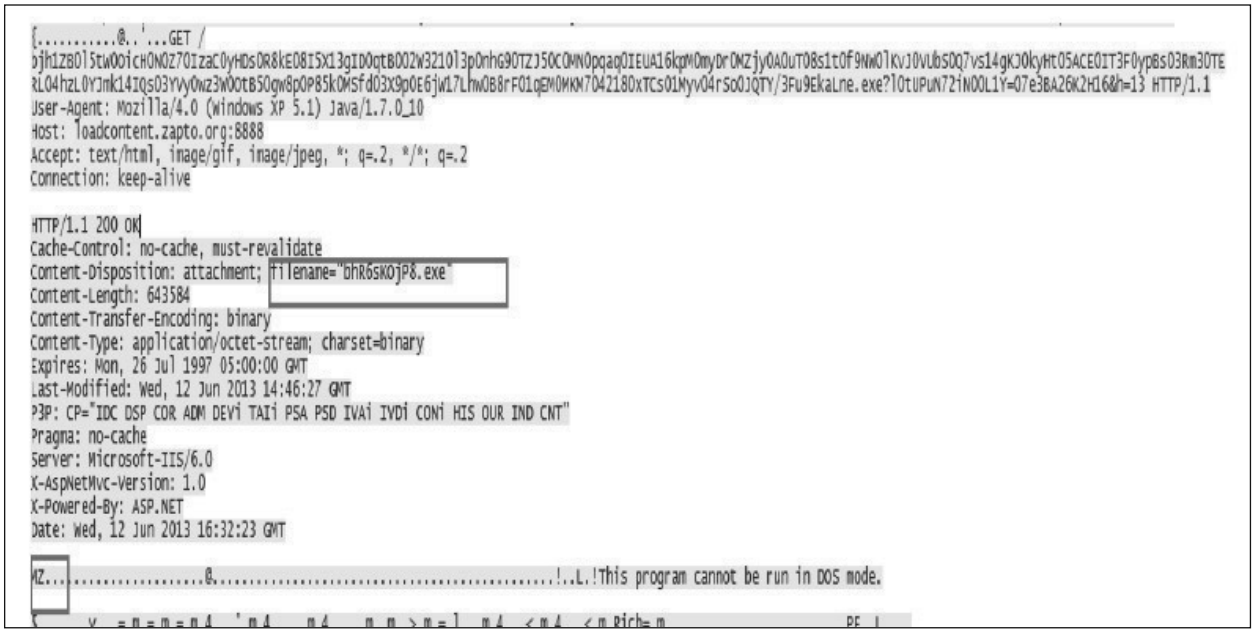

Figure 4: Downloading malicious executable.

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.

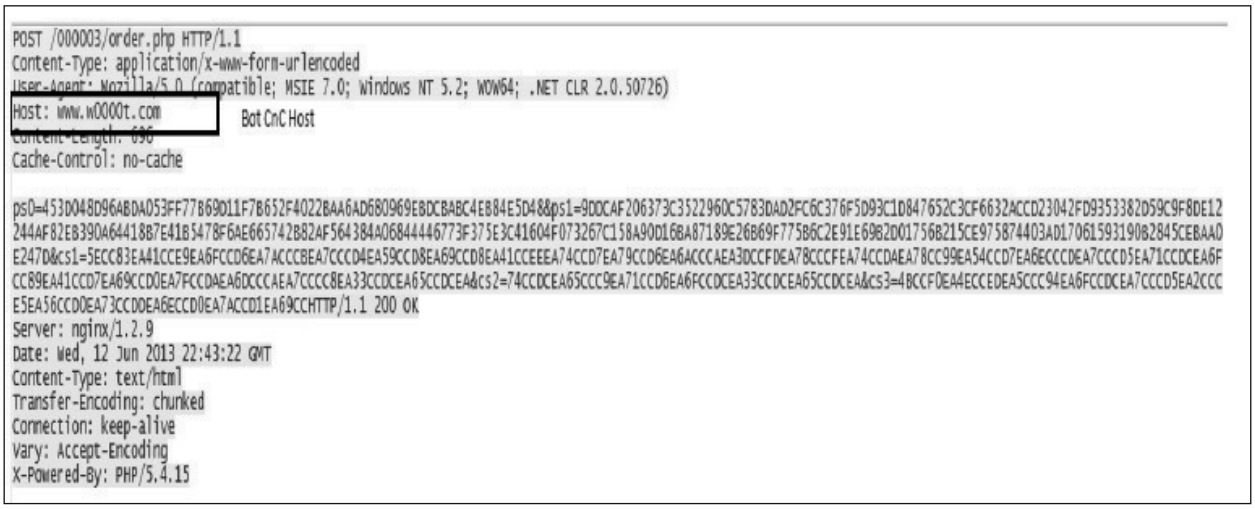

Figure 5: Bot (malware) communicating with C&C.

```
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