

# Malware Analysis Challenge #6

Neil Desai

1. Are these files packed? If so, which packer?

No these files are not packed.

2 (a) Without running the applications, identify what the malware can/will do, then (b)run the applications and identify additional details evident when the applications are run.

## **shell**

Through static analysis of the binary and source code the following functions were determined:

1. Executes commands on the server.
2. Download files using HTTP.
3. Denial of Service Attacks-
  - a. Tsunami - Push/Ack Flooder
  - b. PAN - Syn Flooder
  - c. UDP - UDP Flooder (2 types)
4. Kill a process on the server.
5. Connects to the IRC server 'irc.ircnet.net'

## **cmd.gif**

This file is really a PHP script called 'Defacing Tool 2.0 by r3v3ng4ns'. The page gives the attacker the following list of information:

1. Output of 'uname -a'. Each piece of information is listed on a separate line.
2. UID of the process, effective UID of the process and the group ID of the process
3. If the owner of the process has write access to the directory where the 'cmd.gif' file has been placed.
4. IP address of the hacked server.
5. If the following files are installed:
  - a. /usr/X11R6/bin/xterm
  - b. /usr/bin/nc
  - c. /usr/bin/wget
  - d. /usr/bin/lynx
  - e. /usr/bin/gcc
  - f. /usr/bin/cc

If a file is not found then it is not posted to the screen.

6. If PHP has been configured for safe mode.

7. The version of PHP
8. Full path to the directory where the 'cmd.gif' file is located.
  
9. A place to enter a command, choices of how to run those commands, and an area where the output of the commands will be displayed.

*3. Now, using any methods available to you, which changes, if any, will the malware make on the system?*

The malware did not make any changes to the system. The 'cmd.gif' file does give the attacker the ability to write files to the system.

*4. Now, what are the purpose of the malware? Are they related?*

The files are not really related. The PHP script will be placed on the victim web server. This PHP script will give the attacker various levels of access and abilities depending on how the victim system is configured. In this case the attackers choose to download the 'kaiten.c' file. The attackers could have chosen to other things such as add a user, ready any files, etc.

The 'shelll' file can be used for DDoS attacks or run various commands on the victim system. This file can be used independently of the 'cmd.gif' file.

*5. Why didnt the 'shelll' or the 'cmd' applications show up at the ps aux ?*

The 'shelll' file gets forked to a process named '-bash'. This does show up in the output of 'ps aux'. There were four processes named '-bash' in the output.

The 'cmd.gif' file is just a file that is processed by PHP. PHP will be used by whatever web server that is on the victim system.

*6. Do you have any clues of how the machine was compromised?*

From looking at the 'logs-sec.txt' file it looks like this was an ssh brute force attack against the system. We see at 'Feb 13 19:35:23' the sftp subsystem was started. This was most likely to move the 'cmd.gif' and 'shelll' files to the victim system. From the 'ps aux' output there does not seem to be a httpd server running. Also the 'netstat.txt' file does not show any processes listening on port TCP ports 80 or 8080. There is an account called 'wwwrun' for the apache process so this system does seem to have an httpd capabilities, but it does not seem to be running it at the time the various commands were run.

*7. About the 'shelll' and cmd.gif file, what useful information could you get?*

The 'shell1' file is nothing more than a the 'kaiten.c' file compiled to something less noticeable. The file was not statically compiled or stripped. Because the executable was not stripped it makes it a lot easier to identify the actual source code of the executable. The file was most likely compiled on the system. By looking at the output of 'ls -lan shell1':

```
-rw-r--r--  1 1000 1000 29662 2006-02-21 02:30 shell1
```

From this we can see that the UID and GID are compiled with a user account that's numeric values are 1000/1000. The 'backup' account listed in the 'passed.txt' file shows that its UID is 1000. By looking at the 'logs-sec.txt' file we see various username/backword combinations being used. If the attacker had created the 'backup' account then there would be no use for the brute force attack.

The 'cmd.gif' file is a PHP script that is partially written in Spanish. There were two parts that were written in English. This tool makes it easier to do things on the system once it is infected. The reason for the name 'cmd.gif' is because most people will exclude certain files from being searched for. Since there are probably many different image files on the web server a sample search of the logs might look something like:

```
grep -vi (jpg|gif) <log-file>
```

This will look for any file that contains the strings 'jpg' or 'gif' and it is not case sensitive. The administrator will then think that they are only looking at relevant files and miss the actual attacks.

#### *8. Using all your creative mind, please, describe the possible attack scenario... :-)*

The attacker used the ssh brute force attack against the system. At that point they had a bash shell under the 'backup' account. They could have seen what privileges they had at that point and/or used a local privilege escalation attack to gain root access. For some reason they choose to upload a file or files. Because there is no httpd process running I assume that it was not running at the time of the attack so the attacker moved both files to the system. Once the 'cmd.gif' file is uploaded they could have done anything they wanted. They would have been able to bypass any firewall, but they might have been caught by a NIDS depending on how it is configured.

#### *9. Based on this attack, which security measures would you recommend to this linux box owner?*

They should probably not have SSH open to the Internet. If they need it to be open they should limit it via the source IP or using something like portknocking. They should also be running be sending their logs to remote system so that if the attacker wipes the local files they would still have a lot of information. The username/password combinations should be more secure. If they can they should try to use some other form of authentication, like PKI. They

might also look at chrooting their services so that incase someone is able to break in they will have limited access.

## Preparation

Since this file is malware that is designed for Linux I decided to use a combination of cygwin and Debian for my analysis. The cygwin environment will allow us to analyze the file without worrying about getting infected. The Debian environment will allow us to analyze the file's real time interaction with the infected host. (NOTE: All static analysis is done using cygwin.)

The first thing is to download the file and make sure that it did not get corrupt.

```
$ wget http://handlers.sans.org/pbueno/shell11.zip
--21:37:18-- http://handlers.sans.org/pbueno/shell11.zip
      => `shell11.zip'
Resolving handlers.sans.org... 82.165.161.37
Connecting to handlers.sans.org|82.165.161.37|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12,767 (12K) [application/zip]

100%[=====
=====>] 12,767

21:37:18 (24.60 MB/s) - `shell11.zip' saved [12767/12767]

$ md5sum shell11.zip
d0ae8d57b4c1eb4cc6507fcf0c130883 *shell11.zip
```

The file matches the MD5 checksum on on web site. To see what the file type is we use the '/usr/bin/file' utility.

```
$ /usr/bin/file shell11
shell11: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for
GNU/Linux 2.2.5, dynamic
libs), for GNU/Linux 2.2.5, not stripped
```

This file has not been stripped so there will be a lot of useful information left in the executable which will help when disassembling it. So far there is no indication of any type of code obfuscation (i.e. packer). The next step is to run the '/usr/bin/strings' utility on it. Since there is a lot of output from this I piped the information into a separate file and looked at that file for interesting information. In Appendix A there is a list of interesting strings that were found in the executable.

To get a better information about the executable I used the 'objdump -x' command on the file to look at the headers. This gave us the information about what type of system was used to compile the file and the original name of the file.

## Appendix A

```
/lib/ld-linux.so.2
SuSE
libc.so.6
strcpy
waitpid
vsprintf
recv
connect
atoi
getpid
fgets
memcpy
pclose
sleep
socket
select
popen
accept
write
kill
strcat
bind
strncmp
strncpy
strcasecmp
sendto
bcopy
strtok
listen
fork
irc.ircnet.net
/usr/dict/words
NOTICE %s :GET <host> <save as>
NOTICE %s :Unable to create socket.
NOTICE %s :Unable to resolve address.
NOTICE %s :Unable to connect to http.
GET /%s HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.75 [en] (X11; U; Linux 2.2.16-3 i686)
Host: %s:80
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
```

```

Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
NOTICE %s :Receiving file.
NOTICE %s :Saved as %s
NOTICE %s :Spoofs: %d.%d.%d.%d
NOTICE %s :Spoofs: %d.%d.%d.%d - %d.%d.%d.%d
NOTICE %s :Kaiten wa goraku
NOTICE %s :NICK <nick>
NOTICE %s :Nick cannot be larger than 9 characters.
NICK %s
NOTICE %s :DISABLE <pass>
Disabled
Enabled and awaiting orders
NOTICE %s :Current status is: %s.
NOTICE %s :Already disabled.
NOTICE %s :Password too long! > 254
NOTICE %s :Disable sucessful.
NOTICE %s :ENABLE <pass>
NOTICE %s :Already enabled.
NOTICE %s :Wrong password
NOTICE %s :Password correct.
NOTICE %s :Removed all spoofs
NOTICE %s :What kind of subnet address is that? Do something like: 169.40
NOTICE %s :Unable to resolve %s
NOTICE %s :UDP <target> <port> <secs>
NOTICE %s :Packeting %s.
NOTICE %s :PAN <target> <port> <secs>
NOTICE %s :Panning %s.
NOTICE %s :TSUNAMI <target> <secs>
NOTICE %s :Tsunami heading for %s.
NOTICE %s :UNKNOWN <target> <secs>
NOTICE %s :Unknowning %s.
NOTICE %s :MOVE <server>
NOTICE %s :TSUNAMI <target> <secs> = Special
packeter that wont be blocked by most firewalls
NOTICE %s :PAN <target> <port> <secs> = An advanced syn
flooder that will kill most network driver
s
NOTICE %s :UDP <target> <port> <secs> = A udp flooder
NOTICE %s :UNKNOWN <target> <secs> = Another non-
spoofer udp flooder
NOTICE %s :NICK <nick> = Changes the
nick of the client
NOTICE %s :SERVER <server> = Changes servers
NOTICE %s :GETSPOOFS = Gets the
current spoofing
NOTICE %s :SPOOFS <subnet> = Changes
spoofing to a subnet
NOTICE %s :DISABLE = Disables all
packeting from this client
NOTICE %s :ENABLE = Enables all
packeting from this client
NOTICE %s :KILL = Kills the
client
NOTICE %s :GET <http address> <save as> = Downloads a
file off the web and saves it onto the hd

```

NOTICE %s :VERSION	= Requests
version of client	
NOTICE %s :KILLALL	= Kills all
current packeting	
NOTICE %s :HELP	= Displays this
NOTICE %s :IRC <command>	= Sends this
command to the server	
NOTICE %s :SH <command>	= Executes a
command	
NOTICE %s :Killing pid %d.	
TSUNAMI	
UNKNOWN	
NICK	
SERVER	
GETSPOOFS	
SPOOFS	
DISABLE	
ENABLE	
KILL	
VERSION	
KILLALL	
HELP	
IRC	
NOTICE %s :%s	
MODE %s -xi	
JOIN %s :%s	
WHO %s	
PONG %s	
NOTICE %s :I'm having a problem resolving my host, someone will have to	
SPOOFS me manually.	
PRIVMSG	
PING	
GCC: (GNU) 3.3.5 20050117 (prerelease) (SUSE Linux)	
kaiten.c	