APRIL 2012

virus BULLETIN

Fighting malware and spam

CONTENTS

IN THIS ISSUE

CROSS INFECTOR

There are already at least two known 32-bit and 64-bit cross-infectors for Windows, but W32/W64.Amfibee is the first 32/64-bit cross-infector for Windows that is almost entirely a single block of code. Peter Ferrie explains more.
page 4

GUIDE TO UNPACKING

By packing their malicious executables, malware authors can be sure that when they are opened in a disassembler they will not show the correct sequence of instructions, thus making malware analysis a more lengthy and difficult process. Abhishek Singh provides a quick reference guide for unpacking malware from some of the most commonly used packers.
page 11

FRENCH PHISH

Phishing is a global problem, but there are some geographic variances. Sébastien Goutal presents a study of the phishing attacks typically seen in France.
page 17

ISSN 1749-7027

vb

'The industry seems to be surprised when old attacks are repurposed on new systems.'

Tim Armstrong, Kaspersky Lab

IS ANDROID SIMPLY WINDOWS ALL OVER AGAIN?

It has been very interesting to observe the meteoric rise of the *Android* operating system over the past few years. As we've seen in the past, the more popular a platform becomes, the more cybercrime and malicious activity target it. In 2011, the platform became the most popular target for cybercriminals, with attacks and exploits focusing on financial gain.

What is perhaps more interesting and unfortunate is that many of the attacks on the *Android* platform are not new – the technical nuances may be different, but the premise is the same. I'm not the first to note that successful *Android* attack techniques were first seen on *Windows* years earlier, but the problem continues to grow.

Let's look at the differences. Early *Windows* malware was more of a disorganized nuisance than an effort to make money. This changed fundamentally once criminals released the profit potential. *Android* malware writers entered the field with this knowledge, and we see very little, if any, malware specifically designed to harm the device – there's no money in that.

We have seen mobile malware that can exploit the operating system, gain administrator privileges, install remote access backdoors, install banking malware and join botnets. We've seen fake anti-virus, phishing, adware and spyware.

Yet, the industry seems to be surprised when these old attacks are repurposed on new systems. Why? Aren't mobile devices just one more computer we use? Shouldn't we all have seen this coming?

While some of these attacks are unavoidable, many could have been avoided with better design – design we should have learned about based on the mistakes made in our *Windows* past.

Let's take root exploits for example. With the sheer amount of code involved in designing an operating system, it is impossible to avoid a mistake that could enable an escalation of privilege exploit. In the *Windows* (and *Apple*) world, the response is to provide an update as soon as possible to close the flaw that allowed the attack. It's the same on *Android*, but there is a lack of consistent updates in a timely manner, and a lack of support for older platforms. We need a system of modular security patches for current and (especially) older systems. What we don't need is a new version of the operating system running on shiny new hardware every six months. One could argue that it still takes *Windows* a long time to address such flaws, but they do get addressed eventually. In many cases, if your mobile device has lost support, the flaw will never be addressed.

Perhaps it's the nature of the modern disposable mindset: if a device stops working, you don't fix it, you replace it. Perhaps that's what all the companies that sell *Android* hardware are banking on. However, you can't expect everyone to upgrade to a new device every six months, and you certainly can't do it in the name of security.

Android was designed with security in mind. But it was not designed with users in mind. Take the app permissions screen. Most people click past it as fast as their fingers will allow. While the idea of making permissions known to the end-user is a good idea, the *Windows* installer screen has taught users to click and click until they're done. With the recent spate of adware arriving for *Android*, perhaps it would make more sense to warn users how much of their data is being sent to third parties. *Google* decided not to remove the apps containing the so-called 'Counterclank' advertising because it did not violate its terms of service. Perhaps this is because *Google* is primarily an advertising company. Didn't we already hash out these overly aggressive advertising practices on *Windows*? Why has data leakage become ok just because we're on a new platform?

So, is *Android* simply *Windows* all over again? In some ways, it's worse. Companies are already aware of the threat, and have done little to protect against it. It is not in *Google*'s or the ISP's or even the device manufacturer's fiscal interest to release updates consistently at this point. It is important to sell new devices with new service plans. Until this situation changes (or becomes less profitable), we can expect nothing to change.

NEWS

PAY AND SATISFACTION INCREASE IN IT SECURITY

InformationWeek had some good news for IT security professionals last month when it revealed the results of its 2012 Salary Survey. According to the survey (conducted between November 2011 and January 2012), the median base salary for IT security workers in the US has risen by $7,000 this year, with managers also seeing an increase. Overall job satisfaction (taking into account compensation, benefits, and other employment aspects) also saw an increase for both staff and management.

It will come as little surprise to anybody who has worked in the industry for any length of time (or ever attended a *VB* conference) that women are still in the minority in IT security. A whopping 88% of the IT security workers and 91% of the managers surveyed were male. As seen in other industries, salaries for the female security professionals were below those of their male counterparts, but the results of the survey suggest that the gap may be closing: female workers drew a median base salary of $95,000 (compared with $87,000 last year) while male workers earned $97,000 (compared with $90,000 last year). At management level there was a $5,000 difference between the median salaries for men and women.

The survey also showed that it pays to have certifications. The median base salary for employees with security certifications (such as CISSP, CISA and CISM) was $99,000, compared with $93,000 for those without the letters after their name. At management level, certified managers were shown to be earning a median base salary of $119,000, while their uncertified colleagues took home an average of $14,000 less.

ACADEMIC EXCELLENCE REWARDED

Eight UK universities have each been given a cash injection of £50,000 and awarded 'Academic Centre of Excellence in Cyber Security Research' status by the Government Communications Headquarters (GCHQ).

The institutions recognized by GCHQ were: the University of Bristol, Imperial College London, Lancaster University, the University of Oxford, Queen's University Belfast, Royal Holloway, the University of Southampton and University College London.

The Centres of Excellence are set to run for a period of five years – during which time GCHQ will encourage other universities to develop their capabilities in the area of cybercrime research. It is hoped that this will considerably strengthen the UK's cyber research community and attract the best academics and research students both in the UK and from overseas.

Prevalence Table – February 2012[1]

Malware	Type	%
Autorun	Worm	8.93%
Exploit-misc	Exploit	5.99%
Heuristic/generic	Virus/worm	5.63%
Heuristic/generic	Trojan	5.54%
Crack/Keygen	PU	5.48%
Conficker/Downadup	Worm	5.01%
Sirefef	Trojan	4.67%
Iframe-Exploit	Exploit	4.17%
Adware-misc	Adware	3.16%
BHO/Toolbar-misc	Adware	2.91%
Agent	Trojan	2.67%
Sality	Virus	2.52%
Injector	Trojan	2.40%
Kryptik	Trojan	2.24%
Downloader-misc	Trojan	2.23%
LNK-Exploit	Exploit	1.95%
Crypt	Trojan	1.70%
Blacole	Exploit	1.60%
AutoIt	Trojan	1.53%
Virut	Virus	1.45%
Dofoil	Trojan	1.44%
Freeware-downloader	PU	1.40%
Encrypted/Obfuscated	Misc	1.27%
PDF-Exploit	Exploit	1.26%
JS-Redir/Alescurf	Trojan	1.21%
Redirector	PU	1.20%
Dropper-misc	Trojan	1.14%
FakeAV-Misc	Rogue	1.12%
Backdoor-misc	Trojan	0.85%
InstallCore	Adware	0.84%
VB	Worm	0.84%
Lethic	Trojan	0.83%
Others[2]		14.81%
Total		100.00%

[1]Figures compiled from desktop-level detections.

[2]Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

MALWARE ANALYSIS 1

'AMFIBEE'-OUS VEHICLE

Peter Ferrie
Microsoft, USA

A cross-infector is typically implemented as two viruses stuck together, simply because it's the easiest (and in most cases the *only*) way to do it. However, the x64 technology is a special case – 64-bit instructions use a prefix which corresponds to a register decrement in 32-bit mode. With careful coding, that effect can be reversed as appropriate, and then code can be shared between the 32-bit and 64-bit platforms. There are already at least two known 32-bit and 64-bit cross-infectors for *Windows*, but now we have the first 32-bit and 64-bit cross-infector for *Windows* that is almost entirely a single block of code: W32/W64.Amfibee.

STACKING THE RANKS

The first generation of the virus begins by saving the relative virtual address of the original entry point on the stack. However, depending on the image base value that was used when building it, this value might be completely wrong. In order to account for Address Space Layout Randomization (ASLR), the virus applies the current image base value from the ImageBaseAddress field in the Process Environment Block. This is an interesting way to deal with ASLR – it is more common simply to calculate the difference between a branch instruction and the host entry point. The virus also saves the current stack pointer to a field in its body. Using this value the virus can undo any changes to the stack at any point during the execution of the code. This is particularly important during API resolution, since the virus cannot easily determine how many APIs have been saved before something goes wrong.

DETERMINISM

The virus begins by retrieving the base address of ntdll.dll. It does this by walking the InMemoryOrderModuleList from the PEB_LDR_DATA structure in the Process Environment Block. This is compatible with the changes that were made in *Windows 7*. The required offsets within the InMemoryOrderModuleList are platform-dependent. The virus determines the platform on which it is executing by using a RIP-relative instruction. On the 32-bit platform, the instruction returns a zero in the register. On the 64-bit platform, the same instruction returns an address in the register. The virus also saves the pointer to the current position in the InMemoryOrderModuleList so that it can resume the parsing later to find the base address of kernel32.dll. If the virus finds the PE header for ntdll.dll, it resolves the two required APIs: RtlAddVectoredExceptionHandler and RtlRemoveVectoredExceptionHandler.

The platform determination code is the first 'mistake' in the code. The wrong size of register is checked, resulting in a redundant prefix after every check, which leads to an unnecessary increase in the size of the code. There are a number of similar 'mistakes' in the code.

AN EXPORT EXHORT

The virus uses hashes instead of names, but unlike previous viruses by the same author [1–3], the hashes are not sorted alphabetically according to the strings they represent. As a result, the export table must be parsed repeatedly in order to resolve the APIs. This is not really a problem, it is just not an efficient way to do things. There is also no obvious reason for doing it, since the two registers that are used during the resolution are not altered by anything else. Thus, if the hashes were sorted, the API resolution could easily continue from the current position, with no increase in the size of the code.

Each API address is placed on the stack for easy access, but because stacks move downwards in memory, the addresses end up in reverse order in memory. The virus also checks that the exports really exist by limiting the parsing to the number of exports in the table. The hash table is terminated with a single byte whose value is 0x2a (the '*' character). This is a convenience that allows the file mask to follow immediately in the form of '*.exe'. The virus retrieves the base address of kernel32.dll by fetching the next entry in the InMemoryOrderModuleList list, using the pointer that was saved earlier. The same routine is used to retrieve the addresses of the API functions that it requires, which is the minimum set of APIs that it needs for replication (but because of a bug, more than it actually uses): findfirst/next, open, map, unmap (see below), close.

As with previous viruses by the same author, this virus only uses ANSI APIs. The result is that some files cannot be opened because of the characters in their names, and thus cannot be infected. The virus searches in the current directory (only), for objects whose names end in '.exe'. This is intended to be restricted to files, but can also include any directories that have such a name, and there is no filtering to distinguish between the two cases. For each such file that is found, the virus attempts to open it and map a view of the contents. There is no attempt to remove the read-only attribute, so files that have that attribute set cannot be infected. In the case of a directory, the file open will fail, and the map will be empty. The virus registers an exception handler at this point, and then checks whether the file can be infected.

RELOCATION ALLOWANCE

The virus is interested in Portable Executable files for either the x86 or x64 platforms. Renamed DLL files are not excluded, nor are files that are digitally signed. The subsystem value is checked, but incorrectly. The check is supposed to limit the types to GUI or CUI but only the low byte is checked. Thus, if a file uses a (currently non-existent) subsystem with a value in the high byte, then it could potentially be infected too.

The virus checks the Base Relocation Table data directory to see if the relocation table begins at the start of the last section. If it does, then the virus assumes that the entire section is devoted to relocation information. This could be considered to be a bug. The virus checks that the physical size of the section is large enough to hold the virus code. There are two bugs in this check.

The first is that the size of the relocation table could be much smaller than the size of the section, and other data might follow it. The data will be overwritten when the virus infects the file. Further, the value in the Size field of the Base Relocation Table data directory cannot be less than the size of the relocation information, and it cannot be larger than the size of the section. This is because the value in the Size field is used as the input to a loop that applies the relocation information. It must be at least as large as the sum of the sizes of the relocation data structures. However, if the value were larger than the size of the relocation information, then the loop would access data after the relocation table, and that data would be interpreted as relocation data. If the relocation type were not a valid value, then the file would not load. If the value in the Size field were less than the size of the relocation information, then it would eventually become negative and the loop would parse data until it hit the end of the image and caused an exception.

The second bug is that by checking only the physical size and not the virtual size, whatever the virus places in the file might be truncated in memory if the virtual size of the section is smaller than the physical size of the section. Both of these bugs are also present in some of the other viruses created by the same author.

If the section appears to be large enough, then its attributes are marked as executable and writable, and the virus copies itself to the relocation table. After copying itself, the virus zeroes the value in the Offset field of the Base Relocation Table data directory, saves the original entry point in the virus body, and then sets the host entry point to point directly to the virus code.

OFF THE MAP

The virus code ends with an instruction to force an exception to occur. This is used as a common exit condition. However, the virus does not recalculate the file checksum, even though it might have changed as a result of infection. It also does not restore the file's date and timestamps, making it very easy to see which files have been infected, even though the file size does not change.

There is a bug here, the severity of which depends on the configuration of the environment.

The bug is that the virus calls the wrong API when attempting to unmap the view of the file (and therefore the 'unmap' API is never used). The correct index is used to access the API table, but the virus uses the wrong calling convention, so it ends up calling the CloseHandle() API instead. The act of calling the CloseHandle() API with an invalid handle has a particular effect if a debugger is present, and the same effect if a debugger is not present but if a certain registry value contains a certain value. Normally (that is, no debugger and no registry value), the result is simply that an error code is set, and there is no visible effect. This probably explains why the bug was not noticed.

If a debugger is present, then *Windows* will raise an exception. If a debugger is not present, but the FLG_ENABLE_CLOSE_EXCEPTIONS (0x400000) flag is set in the 'HKLM\System\CurrentControlSet\Control\Session Manager\GlobalFlag' registry value prior to the system being rebooted, then an exception will also be raised. Further, if the flag were set while the virus was running, then the virus would be terminated by *Windows* because the virus unregisters the exception handler prior to closing the file. In that case, the bug would have been very noticeable.

However, the bug will be particularly noticeable in a directory that contains many executable files, regardless of their suitability for infection. The problem is that since the map view is never unmapped, it remains in memory – one map per file that is examined. The size of the map is equal to the size of the file, and each of the maps is aligned to a 64KB block. This is in addition to the host image which is also present in memory, along with its associated DLLs. Thus, it might require only a few hundred large files to be mapped before the memory becomes so scarce that the host simply cannot run after the virus completes its work.

SIZE DOES MATTER

The code is mostly optimized for size, but some obvious size optimizations are missing, such as during the transfer of control that appears after the API resolution. The existing

code determines the platform and then jumps through the stack using a platform-specific value. However, the check could have been avoided completely by using a particular single-byte instruction earlier in the code. The result would be a single jump instruction using a value that is common to both platforms.

In other cases, such as in the vectored exception handler, at least part of the two code paths could be merged by using a nice trick whereby the platform check is used to skip just the REX prefix. There are other examples of multiple code paths where a single one could have been used, such as finding the image base of kernel32, or indexing the API table on the stack.

There are also cases where the REX prefix is redundant because of an unexpected register behaviour on the 64-bit platform. Specifically, assigning a value to a 32-bit register results in the upper 32 bits of the corresponding 64-bit register being zeroed. This also results in a bug which, fortunately for the virus writer, does not have an effect because of a compatibility decision by *Microsoft*. The bug is that the virus retrieves the 32-bit RVA of the export table from ntdll.dll or kernel32.dll, but forgets to add the full 64-bit image base prior to using it. As a result, only the low 32 bits are valid, but it just so happens that the image base of both of those DLLs is always in the low 2GB range, and thus the size of the image base never exceeds 32 bits. In the case of kernelbase.dll and a number of other introduced DLLs, on the other hand, the size of the image base does exceed 32 bits. If the virus had attempted to access any APIs from such a DLL, then the bug would have caused a crash. However, since only ntdll.dll and kernel32.dll are used, the REX prefix is not needed to access their memory.

CONCLUSION

A virus that can run its code natively on both 32-bit and 64-bit platforms is a bit like a lungfish that can live in water or on land (but perhaps less ugly). Fortunately, this virus is in the early stages of evolution – however, we can probably expect to see future advances in this technique.

REFERENCES

[1] http://www.virusbtn.com/virusbulletin/archive/2011/09/vb201109-Holey.

[2] http://www.virusbtn.com/virusbulletin/archive/2012/01/vb201201-sig.

[3] http://www.virusbtn.com/virusbulletin/archive/2012/02/vb201202-Svar.

MALWARE ANALYSIS 2

ZOMBIFYING TARGETS USING PHISHING CAMPAIGNS

Aditya K. Sood and Richard J. Enbody
Michigan State University, USA

Phishing has grown exponentially over recent years. In this article we analyse the Google E-Card phishing campaign and its accompanying binary to show how a victim's machine is compromised.

Figure 1 shows the Google E-Card phishing email. We gathered the following information from the email:

- A classic spoofing technique was used to make it appear as if the email had been sent from the address E-cards@google.com.
- The email headers pointed to a Chinese server running a webmail interface on port 80. Further investigation indicated that the server was controlled by the dgds.gov.cn authority. When the DNS was mapped and the Whois records were searched, it was found that the server was hosted somewhere in the 'China Unicom Guangdong province network'.

  When we followed the link contained in the email, we noticed that the server was configured to host the freebonus.exe software package. The IP address (58.254.202.103) failed to resolve to any hostname or DNS name, and although it was included on a *Malware Patrol* blacklist [1], our browsers did not raise any

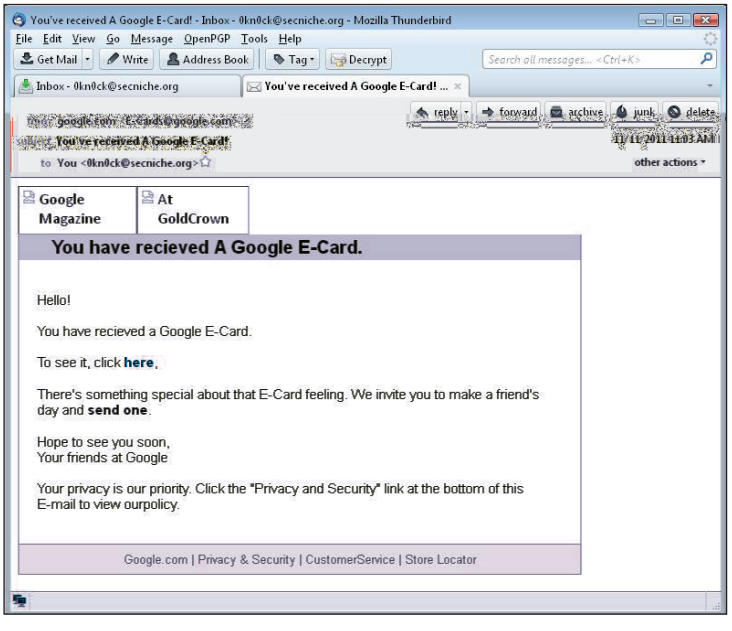

Figure 1: Google E-Card phishing email.

warnings on visiting the site. We also scanned the domain using *Wepawet* [2], which returned benign results with no trace of malicious JavaScript or exploit.

- On reverse tracing the network using a decoy scan against the phishing server, several ports were found to be in an open state, including: FTP (21), SSH (22), SMTP (25), POP3 (110), IMAP (143) and HTTP (80). On querying port 25, the '220 mail.dgds.gov.cn ESMTP Postfix' banner was received, which showed that the mail server was configured for the Chinese government network. The server was not configured as an open relay mailing server, as is usually the case for phishing servers. The IMAP interface was configured to support webmail running on port 80.
- It was found that the SMTP server was configured in a secure manner, with the following commands:

```
220 mail.dgds.gov.cn ESMTP Postfix
EHLO mail.dgds.gov.cn
250-mail.dgds.gov.cn
250-PIPELINING
250-SIZE 52428800
250-VRFY
250-ETRN
250-AUTH PLAIN LOGIN
250-AUTH=PLAIN LOGIN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
```

Even though the VRFY command was enabled, it was not possible to verify the user accounts – the server replied with error message 252 (which states that the server is unable to verify the members of the mailing list). This suggests that either the server is fully compromised or an attack is in progress.

- FTP was running with anonymous access and it was possible to download some files from the server. A custom FTP banner was served when FTP was queried instead of the standard FTP server banner. On fuzzing, the FTP returned a '500 OPS - vsf_sysutil_recv_peek:' error. This error is produced by the VSTFPD server when a capability module is missing from the kernel. However, the server was sufficiently secured not to support the PORT command for launching FTP bounce scans against machines in the same network.

DISSECTING FREEBONUS.EXE

The Chinese server we had traced was serving a zipped self-extracting (SFX) package named freebonus.exe. Some generic techniques were applied to extract the SFX package, but only text files were extracted – which did not seem to make sense from a malware perspective. On closer analysis, we found a number of files that were configured in an obscure manner. As soon as these files were extracted, other critical files were hidden by default. Since the analysis was carried out in a controlled environment we proceeded to consider every step taken by the malware. Once the files were extracted, a generic 'attrib -h *.*' command was run to reveal the files present in the directory. The error received upon running that command was as follows:

```
C:\Documents and Settings\Administrator\Desktop\
freebonus>attrib -h *.*
Not resetting system file - C:\Documents and Settings\
Administrator\Desktop\free
bonus\aliases.ini
Not resetting system file - C:\Documents and Settings\
Administrator\Desktop\free
bonus\away.txt
Not resetting system file - C:\Documents and Settings\
Administrator\Desktop\free
bonus\baby.mrc
Not resetting system file - C:\Documents and Settings\
Administrator\Desktop\free
bonus\control.ini
Not resetting system file - C:\Documents and Settings\
Administrator\Desktop\free
bonus\feel.reg
Not resetting system file - C:\Documents and Settings\
Administrator\Desktop\free
bonus\firefox.exe
Not resetting system file - C:\Documents and Settings\
Administrator\Desktop\free
bonus\fullname.txt
Not resetting system file - C:\Documents and Settings\
Administrator\Desktop\free
bonus\gain.bat
Not resetting system file - C:\Documents and Settings\
Administrator\Desktop\free
bonus\ident.txt
Not resetting system file - C:\Documents and Settings\
Administrator\Desktop\free
bonus\jumbo.ico
Not resetting system file - C:\Documents and Settings\
Administrator\Desktop\free
bonus\lord.mrc
Not resetting system file - C:\Documents and Settings\
Administrator\Desktop\free
bonus\mirc.ini
Not resetting system file - C:\Documents and Settings\
Administrator\Desktop\free
bonus\remote.ini
Not resetting system file - C:\Documents and Settings\
Administrator\Desktop\free
bonus\servers.ini
Not resetting system file - C:\Documents and Settings\
Administrator\Desktop\free
bonus\users.ini
```

The error suggests that the system is not able to reset the files. This error occurs when files are marked with both

hidden (h) and system (s) attributes in the directory. The files can only be retrieved when both flags are removed simultaneously. In order to do this, the 'attrib -h -s *.*' command was run, resulting in the successful extraction of files from the SFX package as shown in Figure 2.

The package was structured in an interesting manner. It was aimed at infecting systems running IRC client software and installed the same set of files as those that are present in a legitimate installation of IRC client software. However, this class of malware has the ability to change the user's machine into a zombie that remains dormant and is only activated when a remote server sends a command. The functionality of the various files are discussed next.

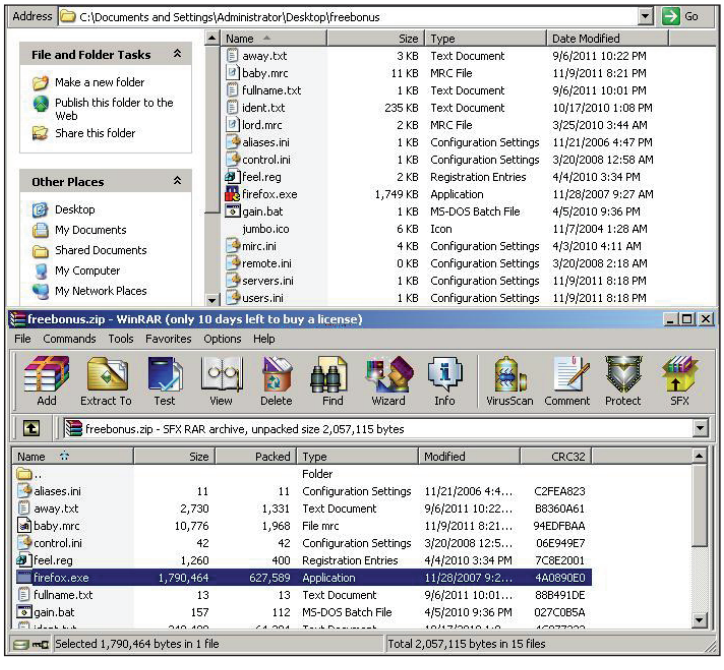

Figure 2: Extracted files from the Freebonus.exe package.

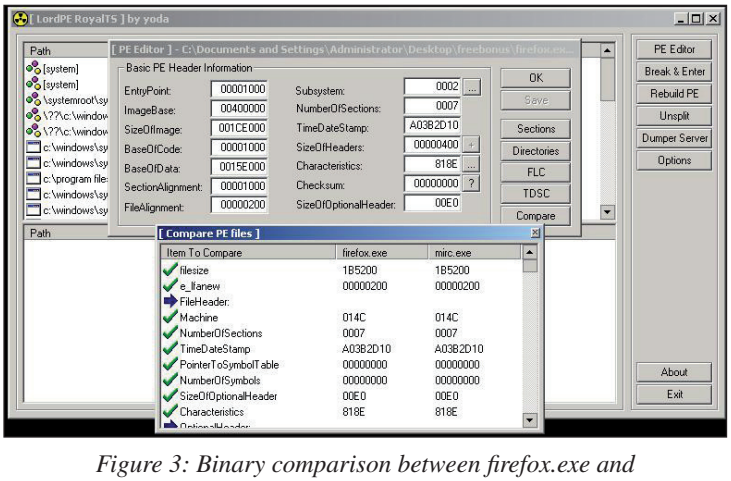

Figure 3: Binary comparison between firefox.exe and mirc.exe.

The main file in the package was 'firefox.exe'. On performing a binary analysis of the file, we found that the executable was written in Borland C, and that the code had the well-defined structure of a message client. This binary looked legitimate in the way it was designed and written. The PE header of 'firefox.exe' gave the impression of being a mIRC client. We wondered whether the malware package was installing the legitimate mIRC client version 6.0.3. In order to verify our hypothesis, we conducted a binary differential analysis. mIRC client version 6.03 was downloaded from the Internet and *LordPE* was used to perform a binary comparison, as presented in Figure 3. We were surprised to find that 'firefox.exe' and 'mirc.exe' were the same in every aspect. This means that the malware package was actually installing a legitimate mIRC client on the victim machine as a service.

Signature-based tools would have raised a false positive on scanning the system. In reality, it is hard to say that an apparently legitimate binary file on the system would turn it into a zombie. The SFX package also contained a number of mIRC scripts. On analysing the mirc.ini file, we found that the IRC client settings had an option defined as hide=1, which directed the IRC client to execute in a hidden manner. The configuration file is shown below:

```
[warn]
fserve=on
dcc=on
[dirs]
logdir=logs\

[about]
version=6.03
show=BR26354

[ports]
random=off
bind=off

[ident]
active=yes
userid=Y
system=UNIX
port=113

[socks]
enabled=no
port=1080
method=4
dccs=no
useip=yes

[clicks]
status=/lusers
```

```
query=/whois $$1 $$1
channel=/channel
nicklist=/query $$1
notify=/whois $$1 $$1
message=/whois $$1 $$1

[dde]
ServerStatus=off
ServiceName=firefox
CheckName=off

[text]
network=All
commandchar=/
linesep=-
timestamp=[HH:nn]
accept=*.jpg,*.gif,*.png,*.bmp,*.txt,*.log,*.wav,*.mid,*.mp3,*.wma,*.ogg,*.zip
ignore=*.exe,*.com,*.bat,*.dll,*.ini,*.mrc,*.vbs,*.js,*.pif,*.scr,*.lnk,*.pl,*.shs,*.htm,*.html
aptitle=Mozilla Firefox
quit=losing my brains
theme=mIRC Classic

[fileserver]
warning=on

[dccserver]
n0=0,59,0,0,0,0
[
[mirc]
user=V
nick=V
anick=V
email=V
host=BudapestSERVER:Budapest.Hu.Eu.Undernet.Org:7000GROUP:Undernet
[
files]
servers=servers.ini
finger=finger.txt
urls=urls.ini
addrbk=addrbk.ini
trayicon=jumbo.ico

[styles]
thin=0
font=0
hide=1
color=default
size=2
buttons=0

[nicklist]
[windows]
main=1244,123,0,34,3,1,0
scripts=-2,1279,-5,931,0,0,0
wchannel=0,610,0,128,0,1,0
wquery=84,610,84,195,2,1,0
wdccs=-1,269,-1,264,0,1,0
wnotify=-1,602,-1,268,0,1,0
playctrl=352,308,178,289,0,0,0

[pfiles]
n0=popups.ini
[notify]

[afiles]
n0=aliases.ini

[rfiles]
n0=users.ini
n1=remote.ini
n2=baby.mrc
n3=lord.mrc
```

The package also contained files such as 'ident.txt', 'servers.ini', 'lord.mrc' and 'baby.mrc'. When the SFX package was unpacked, 'gain.bat' started executing its commands. First, it manipulated the registry entries. Next, it installed the binary into the history folder present in the temporary directory in the '%systemroot' folder. Then it hid the history folder by running the attrib command. Generally, the batch file acted as an installer for the malicious IRC client. The baby.mrc and lord.mrc scripts were executed automatically after the installation of firefox.exe as a service. The malicious firefox.exe client triggered these scripts for joining the remote channel and acting as a zombie for the attacker to control the machine. The mIRC scripts were used to communicate with the admin of the channel by building an ident profile for every server listed in the 'servers.ini' file.

The 'servers.ini' file was used by the malicious IRC client (firefox.exe) for initiating connections to the various IRC servers listed in the file. In order to connect to those servers, the IRC client used the 'users.ini' file to pick up user details. The file contained close to 15 entries related to different IRC servers. The server entries were structured as 'n1=ManaGerSERVER:ff.freebsd.md:8889GROUP:ManaGer', 'n18=BucharestSERVER:82.76.255.62:6662:Undernet', etc. This suggested that IRC servers were differentiated based on the groups. A 'Manager' group was designated for the channel administrators who controlled the bot, while 'Undernet' was the group used for other agents in the network. The IRC servers were found to be in different geographical areas around the globe, which showed that the malware infections were managed in a decentralized manner.

On inspecting 'feel.reg', we found that registry entries were modified for installing 'firefox.exe' as a hidden service. One registry entry, '[HKEY_CURRENT_USER\Software\mIRC\UserName]@="PeNdEjO!"', defined the username of the installed mIRC client as 'PeNdEjO!'. Another entry in the registry was labelled: '[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run] "firefox"="\"C:\\Windows\\temp\\history\\firefox.exe\'. This

caused the process to run in an automated manner when the system was rebooted.

INSIDE MIRC SCRIPTS

The IRC scripts included in the SFX package perform malicious activity on the victim's machine. The following is a snippet of the lord.mrc script:

```
on *:open:?:{
  inc -u3 %msg.chalange 1
  if (%msg.chalange == 2) {
    ame 10Message4 Flood6 detectat2,6 activez4
silence6 pentru4 16 minut2.
    silence +*!*@*
    timerunsilence 1 60 silence -*!*@*
    close -m
  }
}

on *:notice:*:?:{
  if (%notice.chalange.nick != $nick) {
    inc -u3 %notice.chalange 1
  }
  if (%notice.chalange == 2) {
    ame 10Notice4 Flood6 detectat2,6 activez4
silence6 pentru4 16 minut2.
    silence +*!*@*
    timerunsilence 1 60 silence -*!*@*
  }
  set %notice.chalange.nick $nick
}

ctcp *:*:?:{
  if (%ctcp.chalange.nick != $nick) {
    inc -u3 %ctcp.chalange 1
  }
  if (%ctcp.chalange == 2) {
    ame 10CTCP4 Flood6 detectat2,6 activez4 silence6
pentru4 16 minut2.
    silence +*!*@*
    timerunsilence 1 60 silence -*!*@*
  }
  set %ctcp.chalange.nick $nick
}

on *:invite:#:{
  if (%invite.chalange.nick != $nick) {
    inc -u3 %invite.chalange 1
  }
  if (%invite.chalange == 2) {
    ame 10Invite4 Flood6 detectat2,6 activez4
silence6 pentru4 16 minut2.

    silence +*!*@*
    timerunsilence 1 60 silence -*!*@*
  }
  set %invite.chalange.nick $nick
}
```

```
on 1:connect:{
  nick $read ident.txt $+ $r(a,z)
  anick $read ident.txt $+ $r(a,z)
  fullname $read fullname.txt
  identd on $read ident.txt
  .timer 1 5 mode $me +iwx
  .timer 1 7 silence +*!*@*,~*!*@*undernet.org
  .timer 1 17 secure
  .notify on
}

on *:notice:*:#:{ hinc -mu2 spam $chan | if
$hget(spam,$chan) >= 3 { mode $me +d | timerunsilence
1 60 mode $me -d | ame  6Am activat modul 4 +d
6pentru 4 1 6minut din cauza floodului2.

---- Truncated ----
```

The script defines the events as invite, open, notice, ctcp etc. Most of the malicious IRC scripts are written as triggers or events that execute when a particular action is taken. Triggers are defined to automate activity from the IRC client. The generic pattern of a trigger statement is 'on <level>:<event>: { ;Statement block }'. The level is defined as the access level on the IRC channel. The following are explanations of some of the triggers from the malicious IRC scripts:

- The 'open' event is created for all access levels on the IRC channel. The 'inc -u3 %msg.chalange 1' command handles the value in variable %msg.chalange. In this case if the value of %msg.chalange is incremented by one, then after three seconds %msg.chalange will be null. After this, if the required condition is matched then the 'ame' command is executed. The 'ame' command sends a specific action to all channels that the bot is currently on. In this script, the 'ame' command sends a '10Message4 Flood6 detectat2, 6 activez4 silence6 pentru4 16 minute2' message, which defines the flooding activity to be started by the bot connected on a particular channel when final notification is sent by the server manager. The command 'silence +*!*@*' hibernates the bot on the channel, and 'timerunsilence' defines the time period for activating the bot on the channel.
- Other triggers include the 'invite' and 'notify' events. The file also contains a 'CTCP' (client-to-client protocol) trigger. The CTCP command is used to perform client-specific functions on the IRC network. CTCP is used widely for operations such as setting a file server on the victim machine or enabling bots to perform operations without user interaction with the IRC client. The CTCP trigger notifies the channel that a victim's machine ($nickname) is open and already established on the communication channel. The generic CTCP command is used as '/ctcp <nickname><ping|finger|version|time|userinfo|clientinfo>'. The 'invite' trigger

is used to invite other users to the same channel. The user list is provided in the users.ini and ident.txt files.

- The 'connect' event is triggered for initiating connections using the ident profile, the IRC client startss and identd server on port 113 on the victim's machine. The 'nick' command reads an entry from ident.txt and starts connecting back to the IRC server silently.
- The final trigger is the 'notice' event that is used to send a specified notice to the user (nick) on the channel. In this script, messages related to spam are sent in a timely manner.

The malicious scripts are sending notifications for starting flooding and spamming activities on the channel.

DISCUSSION

This malware uses IRC scripting to perform malicious activity on victims' machines. Our analysis and evaluation has indicated that IRC scripting is not a very clear programming language. The IRC clients and IRC scripts are designed to activate backdoors on the victim machine by downloading other malicious programs from remote servers. In this sample, the group leader can use the IRC scripts to control the IRC client and force it to connect to predefined IRC servers and join specific channels. While carrying out background research, we found that a similar variant of this malware [3] has previously been analysed.

CONCLUSION

We have analysed the details of a phishing zombie to understand the propagation and distribution of the malware. We found that tracking the malware domain back to its source can provide a wealth of information to better understand the mechanisms. The malicious binary was also dissected to understand the design of this malware that infects machines and turns them into zombies. One aim of this study is to present a glimpse into the methodology used to track back malicious servers for gathering details about the malicious tools.

REFERENCES

[1] http://www.malware.com.br/cgi/submit-agressive?action=list&type=agressive.

[2] Wepawet. http://wepawet.iseclab.org/index.php.

[3] Client-IRC.Win32.mIRC.603, Backdoor.IRC.Zapchast.zwrc. http://www.threatexpert.com/report.aspx?md5=c0d2abe80f901502fb3e7a40f8bf77aa.

TUTORIAL

QUICK REFERENCE FOR MANUAL UNPACKING

Abhishek Singh
FireEye, USA

Malware authors utilize packers to make it difficult for their malware to be reversed – the packers encode the original instructions. By packing a malicious executable, its author can be sure that when it is opened in a disassembler it will not show the correct sequence of instructions. Packers add some instructions at the top of the binary to unpack the executable. The process of decryption is performed in memory at run time, and the state of the application is restored. Since packers work on a compiled executable, the unpacking module must be independent of the original application.

One of the methods that can be used to locate the original entry point (OEP) of the file is to apply break points on the following APIs:

```
GetLoadLibraryA
GetVersionExA
GetEnvironmentA
LoadLibraryA
GetProcAddress
IniHeap
```

These APIs are called by the packers' start-up routines in order to set up the execution environment. When a breakpoint is applied to these routines, we are close to the OEP. When the break point triggers, we can use step-by-step tracing to locate the initialization of the stack frame. The start of the function can be recognized by the initialization of the stack frame.

```
push  ebp
mov   ebp, esp
```

The instructions shown above denote the start of the stack frame. Once these instructions are located, the debugged process can be dumped to obtain the unpacked version of the file.

In the following sections we describe some common packers and the assembly instructions that can be used to locate the OEP.

MANUAL UNPACKING

The purpose of this section is to provide a quick reference guide that will assist malware analysts in the unpacking of malware and reduce the response time for malware analysis – the full technical details of each packer have therefore been omitted.

ASPack

ASPack is an advanced *Windows 32* executable compressor capable of reducing the file size of 32-bit *Windows* (*95/98/ME/NT/2000/XP/2003/Vista/7*) programs by as much as 70%. It is also used by some hackers to protect their programs.

To unpack ASPack, follow the first jmp, and follow JMP EAX. Later in the code you will find the following instructions:

```
mov eax,1
retn 0C
push 0
retn
```

Once these instructions have been identified, as shown in Figure 1, a break point should be put on RETN. When the break point triggers, we are at the OEP. The process can be dumped at this stage, leaving us with the unpacked executable.

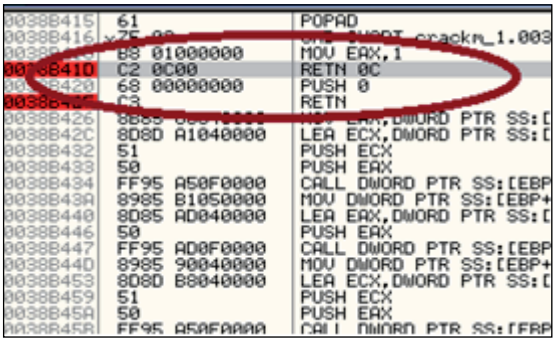

Figure 1: Instructions before the code is unpacked.

OllyScript code for the automatic unpacking of ASPack is shown in Figure 2. The instruction 'findop eip, #6800000000#' locates the PUSH 0 instruction in a debugged process packed with ASPack. Once this instruction is located, the debugger steps once to reach the RETN instruction. The debugger then steps again to reach the OEP instruction. Once the OEP instruction is located the debugger steps once more to reach the OEP. The debugged process can now be dumped to get the unpacked version of the file.

```
findop eip, #6800000000#
go $RESULT
sti
sti
msg "OEP is found for ASPack "
run
```

Figure 2: OllyScript code used to locate the OEP for ASPack.

KKrunchy

KKrunchy [1] is a small executable packer intended for 64k intros. It does not try to pack DLLs and cannot handle exports or TLS. It performs a transform on the input code to allow it to compress better. It will fill uninitialized data sections with zeros and then pack them together with the rest of the code. KKrunchy is often used by malware authors to prevent AV analysts from reversing their code.

In order to unpack KKrunchy, put a break point on LoadLibraryA. When the break point triggers, step the debugger and search for the initialization of the stack frame. Once the stack frame initialization is complete, dump the debugged process. The dumped process is the unpacked version of the executable.

PECompact v2.x

PECompact [2] is fully compatible with DEP and code signing, and provides support for *Windows 7* and *Windows 2008*. It provides a good compression ratio compared to other compressors such as ASPack. The PECompact [3] loader consists of three components. The first is the SEH entry, which transfers control to the second component, the loader decoder. The loader decoder decodes the code and invokes the third component, the primary loader. The loader decoder is stored in the last section (or the second-to-last section if relocations have been preserved). The primary loader exists in uncompressed form at runtime in dynamically allocated memory. To hide the transfer of control, an SEH frame is set up and there is an exception. The exception handler then modifies the code at the exception address to a JMP and continues execution.

Figure 3 shows PECompact's exception handler. The instruction sequence 'PUSH EAX, PUSH DWORD PTR FS:[0], MOV DWORD PTR FS:[0], ESP' sets up the SEH frame. The instruction 'XOR EAX, EAX' sets the value in EAX to zero. The instruction 'MOV DWORD PTR DS:[EAX], ECX' triggers the exception.

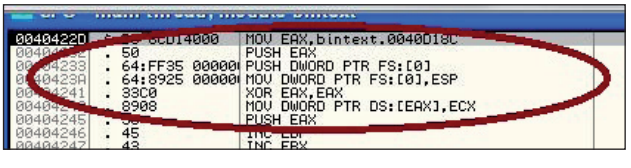

Figure 3: The PECompact exception handler.

To unpack PECompact, follow the exception and step through the code until the instructions shown in Figure 4 are observed. JMP EAX is the jump to the OEP.

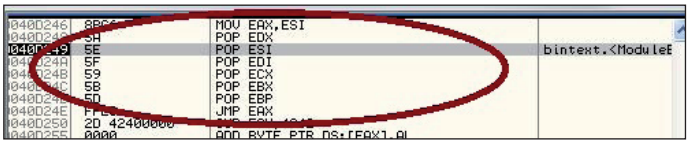

Figure 4: PECompact instructions before unpacking.

Set a break point on JMP EAX, step once, and observe the initialization of the stack frame as shown in Figure 5. Dump the process. The dumped process will be the unpacked executable.

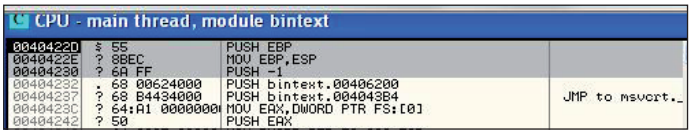

Figure 5: Initialization of the stack frame.

The logic shown in Figure 4 can be converted into script such as that shown in Figure 6 (the script is available from Open RCE [4]).

```
sto
sto
sto
sto
sto
sto
esto
find eip, #8BC65A5E5F595B5DFFE0#
add $RESULT,08
bp $RESULT
run
bc $RESULT
sto
cmt eip,"This is a OEP!"
msg "OEP found, Dumped and fix IAT now!"
ret
```

Figure 6: OllyScript for PECompact.

The instruction 'find eip, #8BC65A5E5F595B5DFFE0#' locates the instructions 'MOV EAX ESI, POP EDX, POP ESI, POP EDI, POP ECX, POP EBX, POP EBP, JMP EAX'. Once these are located, the script steps once at the JMP instruction and the debugger is at the OEP. The debugged process now can be dumped to obtain the unpacked version of the file.

NSPack

NSPack [5] is capable of compressing EXE, DLL, OCX and SCR files. It also has the ability to compress 64-bit executables. It provides support to compress files packed by other packers such as UPX, ASPack and PECompact. It supports direct compression of directories or multiple files. This packer is quite commonly used by malware authors.

As shown in Figure 7, the packer starts with the instructions PUSHFD, PUSHAD.

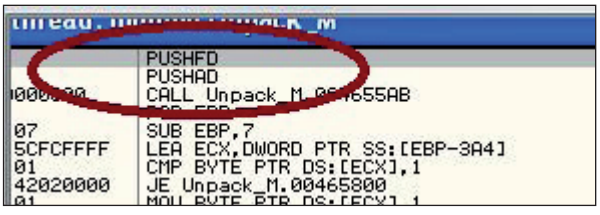

Figure 7: The starting instructions for NSPack.

Check for equivalent POPAD and POPFD instructions, as shown in Figure 8. The JMP instruction follows. Put a break point on the JMP instruction. When the break point triggers, step once and dump the process to obtain the unpacked file.

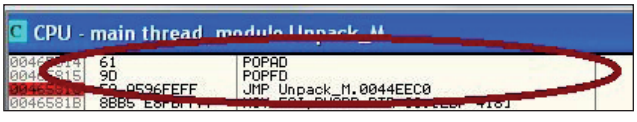

Figure 8: NSPack instructions before unpacking.

The abovementioned logic can be converted into the OllyScript shown in Figure 9. The instruction 'find eip, #619DE9#' locates the instruction POPAD, followed by POPFD, followed by a JMP instruction. Once these are located, the code is debugged, step by step, until the JMP instruction is executed – the debugger has then reached the OEP instruction. By using a plug-in like OllyDump, the process can be dumped to obtain the unpacked version of the file.

```
sti
find eip, #619DE9#
go $RESULT
sto
sto
sto
msg "OEP for NSPack "
```

Figure 9: OllyScript used to locate the OEP for NSPack.

FSG 1.33

FSG stands for Fast Small and Good, and is currently used to pack various malware. It was originally created to pack assembly demos. Since it has a small loader, it is one of the most desirable packers for small executables.

In order to obtain the unpacked executable file for FSG 1.33, put a break point on the LoadLibraryA function, as shown in Figure 10.

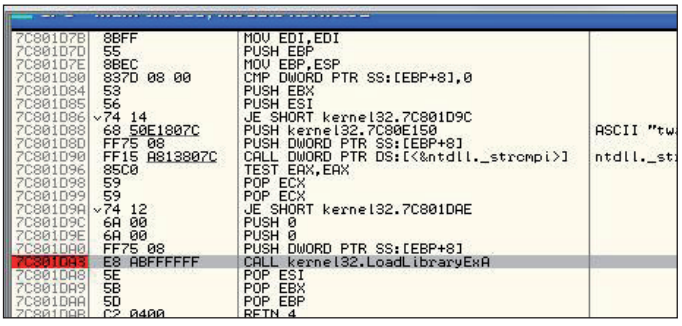

Figure 10: The LoadLibraryA function in FSG 1.33.

When the break point triggers, step a few instructions below until the following instructions are seen:

```
dec byte ptr [esi]
jz xxxxxxxx
PUSH ESI
PUSH EDP
CALL DWORD PIR D5:[EBX=4]
```

When JE Address triggers, we can observe the initialization of the stack frame. We are at the OEP, so dump the process to get the unpacked version of the file.

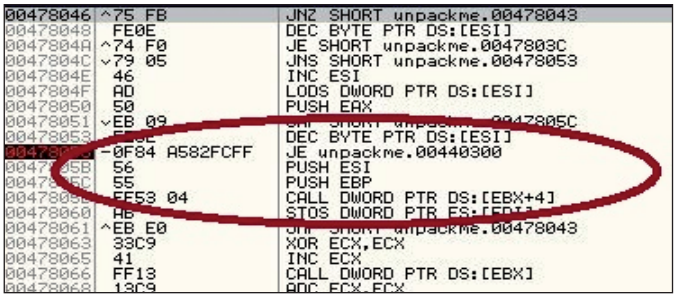

Figure 11: Instructions denoting the end of FSG.

FSG 2.0

For version 2.0 of the FSG packer, the instructions that indicate the end of the FSG stub are as follows:

```
move eax (edi)
inc eax
js address
jnz address
jmp dword ptr [ebx+0Ch]
```

In order to manually unpack a file packed with FSG 2.0, put a break point on LoadLibraryA and execute the compressed file. When it breaks, clear the break point and execute until return (Ctrl -f9). Step through the debugged application until the instructions shown in Figure 12 are reached.

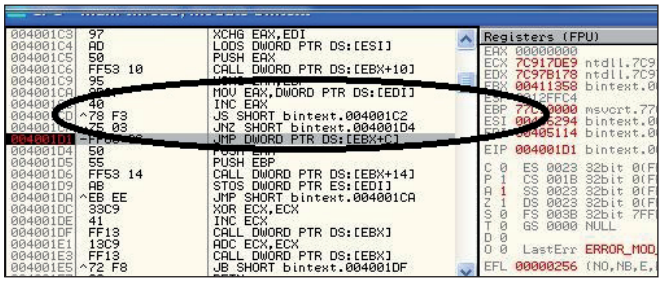

Figure 12: Instructions reached before unpacking FSG 2.0.

Here, 'JMP DWORD PTR DS: PTR [ebx+0Ch]' is the jump to OEP. Once the JMP instruction is executed, dump the process to get the unpacked version of the file.

UPX

UPX [6] stands for Ultimate Packer for eXecutables. It offers an excellent compression ratio which is better than WinZip, Zip and GZIP. It also maintains a checksum for both compressed and uncompressed files. It uses compression algorithms like UCL [7]. UCL has the inherent advantage that the decompressor can be implemented in a few hundred bytes of code. Many malware families such as Qakbot are packed using UPX. It offers very fast compression and decompression speeds: ~10MB/s on a *Pentium* 133. It also offers support for LZMA compression and has support for BSD. LZMA decompression is disabled on the 16-bit platform due to the slow decompression speed on older platforms. It also provides support for two types of decompression routines. The first is the in-place technique, which decompresses the executable in memory. In-place decompression is possible only for some platforms. The extraction of a temporary file, even though it uses extra overhead, allows any executable file format to be packed.

In order to unpack UPX using a manual approach, the end of the UPX routine must be identified. The end of the UPX routine can be identified by the instructions CALL, POPAD and JMP, as shown in Figure 13. Put a break point on the JMP instruction. The JMP instruction will lead to initialization of the stack frame. After the JMP instruction has executed, dump the process by using a plug-in such as OllyDump, and the program is unpacked.

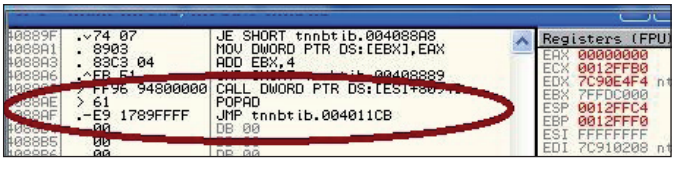

Figure 13: UPX end of routine instructions.

```
var BPforPOPAD
var BPforJMP
findop eip, #61#
mov BPforPOPAD, $RESULT
bp BPforPOPAD
run
findop eip, #E9????????#
mov BPforJMP, $RESULT
bp BPforJMP
run
sti
msg "OEP for UPX. Dump the Process"
```

Figure 14: The OllyScript used to unpack UPX.

The script shown in Figure 14 is the implementation of the logic used to locate the OEP. The instruction 'findop eip, #61#' locates the assembly instruction POPAD, sets a break point on it, and then executes the code packed with UPX. Once the break point is triggered, the instruction 'findop eip, #E9????????#' locates the JMP instruction and sets a break point on it. When the break point triggers, the debugger steps once in the code and is at the OEP. The debugged process can be dumped to get the unpacked version of the file.

PEDiminisher

PEDiminisher is a simple PE packer. It uses the aplib compression/decompression library. Many AV engines have the ability to unpack files packed with PEDiminisher to check for malicious content.

The end routine for PEDiminisher is shown below:

```
pop   EBP
POP   EDI
POP   ESI
POP   EDX
POP   ECX
POP   EBX
JMP   EAX
```

For unpacking, the end instructions must first be located in the packed file (as shown in Figure 15). JMP EAX is the jump to the OEP. Set a break point at the JMP instruction, step once and then dump the process to get the unpacked version of the file.

The instruction 'find eip, #5D5F5E5A95BFFE0#' locates the instructions 'POP EBP, POP EDI, POP ESI, POP EDX, POP ECX, POP EBX, JMP EAX'. The script then steps through the debugger until it reaches JMP EAX. Once it is at JMP EAX, the code steps once and is at the OEP. The OllyDump plug-in can be used to dump the process and we are left with the unpacked version of the executable file.

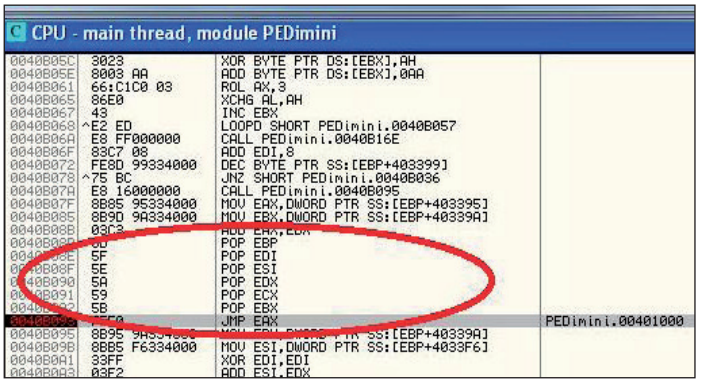

Figure 15: The end instruction for PEDiminisher.

```
find eip, #5D5F5E5A595BFFE0#
bp $RESULT
run
sti
sti
sti
sti
sti
sti
sti
msg "OEP found for PEDiminisher, Dump now!"
```

Figure 16: The OllyScript used to unpack PEDiminisher.

MEW

MEW [8] is an executable tool which was designed to handle small files. It works on 32-bit workstations and uses the LZMA algorithm. It strips reloc tables, Delphi resources, and unused resources. Even though it was designed to handle small files, it can compress large files as well.

The last instruction in the MEW stub, as shown in Figure 17, is RETN. After this instruction a jump to the OEP takes place. Set a break point on the RETN instruction. When the break point is triggered, as shown in Figure 17, step once and then dump the process to get the unpacked version of the file.

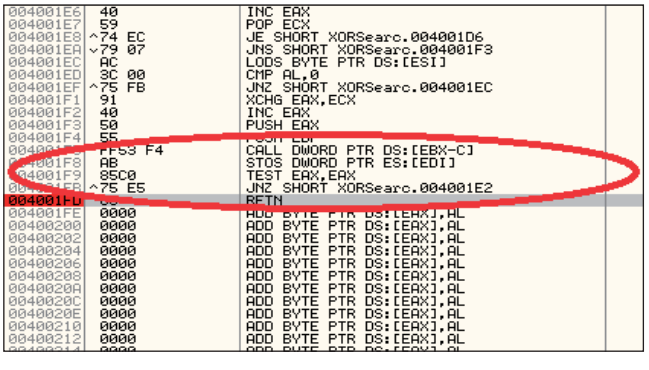

Figure 17: The last instructions for the MEW packer.

```
sti
findop eip, #C3#
go $RESULT
sto
sto
msg "OEP found for MEW"
```

Figure 18: The OllyScript used to unpack MEW.

The logic used to locate the OEP for MEW is shown in Figure 18. The code 'findop eip, #C3#' locates the RETN instruction in the debugged process packed with the MEW packer. Once the RETN instruction is located, the debugger steps once and is at the OEP. The OllyDump plug-in can be used to dump the process and we are left with the unpacked version of the executable file.

CONCLUSION

Reducing the time it takes to perform malware analysis is very important. For static analysis of malware it is important that the malware is unpacked. There are many approaches to unpacking a piece of malware – for example, it can be executed in a virtual environment and then we can capture a memory snapshot of the executing malware. Once we get the snapshot, we can dump the unpacked malware directly from memory. However, it is possible that not all of the code of the unpacked malware will be in memory, so dumping a process from memory might not be an effective unpacking method. Loading a packed malicious executable and executing step by step instructions in a debugger is one of the best ways to locate the OEP and execute the malware. In this article we have provided assembly instructions for the most commonly used packers which can be used to quickly unpack malware. We have also provided OllyScripts for the logic to manually unpack the malware. This can further aid in reducing response time for malware analysis.

REFERENCES

[1] http://www.farbrausch.de/~fg/kkrunchy/.

[2] http://pecompact.com/pecompact.php.

[3] http://www.bitsum.com/pec2av.htm.

[4] http://www.openrce.org/downloads/details/156/PECompact_v.2.40_-_OEP_finder.

[5] http://nspack.download-230-13103.programsbase.com/.

[6] http://upx.sourceforge.net/.

[7] http://www.oberhumer.com/opensource/ucl/.

[8] http://www.softpedia.com/get/Programming/Packers-Crypters-Protectors/MEW-SE.shtml.

FEATURE

FRANCOPHILE PHISHERS

Sébastien Goutal
Vade Retro, France

Phishing is a major threat to email users. Not only do victims face financial loss, but phishing can also result in a loss of trust and confidence in the organizations targeted in the attacks (ISPs, banks, social networks etc.). In this article we present a study of the phishing attacks that we typically see in our home market of France.

PHISHER PROFILE

In our experience, phishers tend to be aged between 15 and 25 years old. They operate alone or in small groups, have relatively limited skills[1] and work on a small scale, using several hacked servers and hosted services.

In contrast, a typical spammer is a professional criminal with significant technical skills[2] and access to many resources. Spammers maintain large networks and send email in very high volumes. While a phisher will send at most a few tens of thousands of emails per day, a botnet has the capacity to send hundreds of millions of emails, and even up to several billion on a daily basis (for instance, the Rustock botnet has the capacity to send 30 billion emails per day).

Phishing provides a quick return on investment, and it requires few resources. The main challenge for phishers is converting money from online accounts into hard cash.

PHISHING IN FRANCE

There is a persistent phishing threat in France that requires daily vigilance. Phishers operate mainly from Morocco (Rabat, Casablanca), and occasionally from Tunisia and France. Why Morocco and Tunisia? First, because the French language is widely spoken there, and secondly, these countries are located outside the European Union – where the risk of prosecution for damages caused in the European Union (and France in particular) is extremely small.

Every day, we identify between 10 and 20 new phishing campaigns or variations of existing ones.

The size of campaigns varies from a few hundred emails to tens of thousands of emails. They last for anywhere between one hour and several days. The average duration of the phishing campaigns we see is about seven hours, with a low and constant intensity.

Globally, there is a low volume of phishing: phishers attempt to keep their activity below a certain threshold in order to remain undetected for as long as possible. This is consistent with the phishing business model which requires a low volume to be profitable.

PHISHING PROCESS

Phisher's infrastructure

The phishing process is rather simple. First, the phisher needs at least two servers: one is dedicated to the dispatch of emails, and another hosts the phishing sites. In most of the cases that we see, phishers use hacked servers[3], but it has also become common for phishers to use hosted services[4].

Templates for phishing sites are widely available on the Internet, however most phishers create their own pages, basing their design heavily on the official site of the targeted company. They always include a script that will send notifications of captured data to one or more mailboxes – typically free webmail addresses such as *Yahoo!* or *Gmail*. This is the safest and easiest way to retrieve stolen credentials; storing them locally on the servers is risky in case the servers are forced to shut down due to abuse reports.

Phishing campaign

Next, the phisher has to compose a phishing email and send it to selected recipients. Previously, phishers compiled emails that were designed to resemble the legitimate emails sent by the target companies – including the company's logo and layout – and the From and Subject headers, as well as body content, were explicit: their primary objective was to convince the majority of end-users that this was a legitimate email. Nowadays, regular phishing is still explicit[5] but less close to the genuine emails sent by the targeted companies: the primary objective is now to bypass spam filters, whose efficiency against phishing has increased.

The phisher must make a choice: write a 'genuine'-looking phishing email that will trap a lot of end-users but will quickly be detected by spam filters, or write an obfuscated email that will trap fewer end-users but will be more difficult to detect. There is a growing tendency towards

[1] Knowledge of a scripting language and a basic knowledge of *Apache*, HTTP and SMTP protocols is more than sufficient to conduct phishing campaigns.
[2] A spammer has to conduct vast malware campaigns to build spambot networks, handle complex peer-to-peer interaction between botnet elements, and monitor an infrastructure of significant size.
[3] For instance, a lot of servers hosting misconfigured or vulnerable versions of *WordPress* are used by phishers to host phishing sites. They can easily be identified by the ' wp-' prefix in the phishing link.
[4] However, the rental of hosted services requires prior theft of credit card credentials in order to anonymize the financial source.
[5] Although this does not prevent phishers from using classic spam techniques to bypass spam filters, such as misspellings or character substitutions.

Figure 1: This phishing scam contains various misspellings. Notice the contrast between the logo that will draw the recipient's attention, and the misspelled words 'Caiise d'Epargne'.

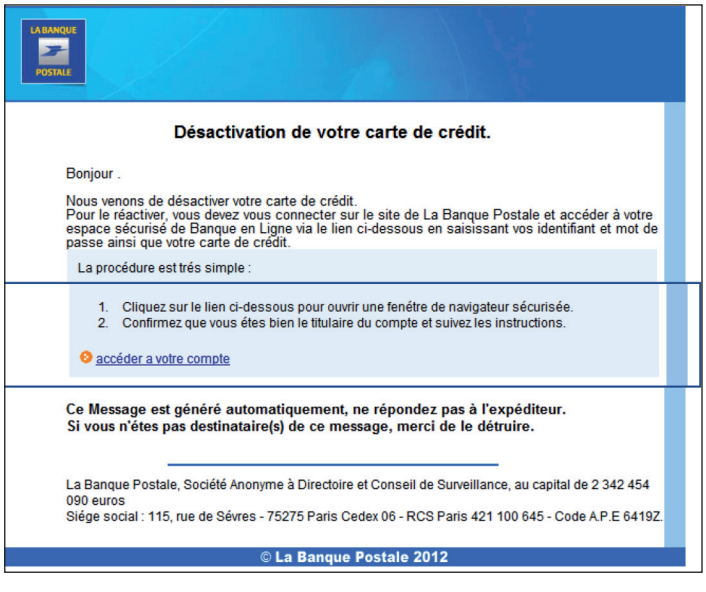

Figure 2: Phishing emails are often deliberately misspelled to bypass spam filters that may be triggered by certain keywords. However, this one is particularly well written: this is an image phishing scam, thus there is no need for misspellings.

extreme obfuscation. For example, some phishing emails contain nothing but a link to the fraudulent site, which make them very difficult to detect in a predictive way.

We see the same arguments being used repeatedly in phishing campaigns to convince the end-user to take action:

- suspicious connection attempts to your account
- unusual transactions on your account
- an unauthorized transaction on your account
- a transaction error
- account or credit card suspended
- new message received
- new invoice received.

It is also very common for phishing emails to request confirmation of customer credentials in order to enhance security: phishers use and abuse the fact that a lot of people are already aware of phishing scams.

Finally, before launching the phishing campaign, the phisher sends one or several phishing emails to the targeted ISP, using an email address hosted by the ISP. By doing this, he can make sure that the phishing campaign will not be blocked from the very beginning. He can then launch the campaign, and start to harvest the stolen data.

PHISHER BUSINESS MODEL

We often see stolen credit card credentials being used to fund online credit accounts, such as *PayPal* and *Ogone*. The credit is used for various operations:

- Purchase of high-value consumer goods (laptop computers, tablet computers, consoles etc.), which are then resold via classified ad websites[6].
- Purchase of plane tickets, which are then resold at lower prices.
- Purchase of hosting services to perpetuate the phishing business.

As money laundering is a complex process and requires close acquaintance with criminal organizations, some phishers also sell credential databases on the black market. These may be used for other activities such as identity theft.

The obvious limitation to the business model is the difficulty of converting money from online accounts into hard cash.

[6] *Leboncoin.fr* is the most popular classified ad website in France, and as such it is widely used for the resale of goods that have been obtained illegally. For this reason the site is under constant surveillance by law enforcement. *eBay* is also widely used for this purpose.

PHISHING TARGETS

Typically, we see the following targets of phishing campaigns:

- Internet service providers
- Banks (*La Banque Postale*, *BNP Paribas*, *Crédit Agricole*, *Crédit Mutuel*, *Caisse d'Epargne*, *Société Générale*) and companies providing financial services (*Visa*, *MasterCard*, *PayPal*, *Western Union*)
- Public services (income taxes, Caisse Primaire d'Assurance Maladie, Caisse d'Allocation Familiale, Electricité De France, Gaz De France)
- Social networks (*Facebook*, *Twitter*)
- Online games (*battle.net*, *RuneScape*)
- Online auction services (*eBay*).

In France, phishers focus heavily on ISPs, because they can easily determine the victim's ISP by extracting the domain name of their email address. *PayPal*[7] and *Visa/MasterCard* are also very highly targeted.

Local banks are also targeted. The phisher does not know the identity of the victim's bank, and there are around ten major banks in France – the probability that the victim is a customer of any of these banks is around 10%. Despite this, phishers still focus on local banks as customers tend to trust them, and are thus more likely to be trapped.

We rarely see phishers targeting foreign banks (e.g. *Bank Of America*, *Chase*, *Wells Fargo*, *Santander*, *Halifax*, *Barclays*, *HSBC* etc.), instead focusing their activities on French banks to maximize their return on investment.

CONCLUSION

Nowadays, phishing is one of the most problematic threats in mail security: in contrast to spam, phishing campaigns are low-volume, targeted, transient, and thus both very difficult to detect in a feedback loop and almost non-existent in honeypots. Moreover, phishing is very profitable for the attackers and can be extremely damaging for the victims.

Two layers of security are generally set up to limit the damage caused by phishing. First, ISPs and ESPs implement message filtering to detect and reject phishing emails. Secondly, most popular Internet browsers provide anti-phishing features[8], the majority of which are based on live phishing URL database queries, to prevent access to forged websites. However, in both cases, the phishing attack first has to be detected and reported, and the delay in this process means that the initial victims go unprotected[9].

It is inevitable that, whatever the sophistication of the implemented protection technologies, phishers will manage to steal valid credentials and will try to use them. We must therefore limit damages by strengthening controls at the point at which these credentials will be used.

E-commerce websites should detect suspicious behaviour – such as multiple payment attempts with different credit card details – and should reject, log and report such attempts.

In addition, banks and online financial services should implement two-factor authentication. Two-factor authentication is an approach to authentication which requires the presentation of two or more of the three authentication 'factors' (something the user *knows*, something the user *has*, and something the user *is*). A classic approach is to send a confirmation code to a mobile phone (something the user *has*), and thus validate authentication. Widespread use of two-factor authentication should limit the damages caused by phishing.

Last but not least, it is essential that there is effective cooperation between authorities in different countries: phishing is an international threat, and we know that phishers take advantage of weaknesses in international cooperation and often carry out their actions with a sense of impunity[10].

[7] The number of *PayPal* accounts is estimated to be over 230 million. It is therefore very likely that the recipient of a phishing email has a *PayPal* account.

[8] Such as *Google Safe Browsing* in *Google Chrome* and *Mozilla Firefox*.

[9] In the best case, it takes several minutes before a phishing campaign is reported by a user in a large ISP feedback loop. We estimate that the time taken to broadcast the signature for a new phishing campaign is between 20 minutes and several hours after the start of the campaign.

[10] For example, it is not unusual for phishers to post about their exploits (with screenshots as evidence) on their *Facebook* walls.

END NOTES & NEWS

SOURCE Boston 2012 will be held 17–19 April 2012 in Boston, MA, USA. For further details see http://www.sourceconference.com/boston/.

The 3rd VB 'Securing Your Organization in the Age of Cybercrime' Seminar takes place 19 April 2012 in Milton Keynes, UK. Held in association with the MCT Faculty of The Open University, the seminar gives IT professionals an opportunity to learn from and interact with top security experts and take away invaluable advice and information. See http://www.virusbtn.com/seminar/.

Infosecurity Europe 2012 takes place 24–26 April 2012 in London, UK. See http://www.infosec.co.uk/.

The Sixth Counter-eCrime Operations Summit will be held 25–27 April 2012 in Prague, Czech Republic. For details see http://apwg.org/events/2012_cecos.html.

TakeDownCon Dallas takes place 4–9 May 2012 in Dallas, TX, USA. Other TakeDownCon events take place 25–30 August in Balitmore, MD, and 1–6 December in Las Vegas, NV. For more information about each see http://www.takedowncon.com/.

The 21st EICAR Conference takes place 7–8 May 2012 in Lisbon, Portugal. For details see http://www.eicar.org/17-0-General-Info.html.

The CARO 2012 Workshop will be held 14–15 May 2012 near Munich, Germany. For more information see http://2012.caro.org/.

CONFidence 2012 will take place 23–24 May 2012 in Krakow, Poland. For details see http://2012.confidence.org.pl/virus-bulletin.

EC-Council Summit Boston takes place 4–7 June 2012 in Boston, MA, USA. Other summits take place 11–14 June in San Antonio, CA, and 20–23 August in San Jose, CA. For details of each see http://www.eccouncil.org/training/advanced_security_training/cast_summit.aspx.

The MAAWG 25th General Meeting will be held 5–7 June 2012 in Berlin, Germany. MAAWG meetings are open to members and invited guests only. For questions and invite requests see http://www.maawg.org/contact_form.

NISC12 will be held 13–15 June 2012 in Cumbernauld, Scotland. The event will concentrate on 'The Diminishing Network Perimeter'. For more information see http://www.nisc.org.uk/.

The 24th annual FIRST Conference takes place 17–22 June 2012 in Malta. For details see http://conference.first.org/.

Black Hat USA will take place 21–26 July 2012 in Las Vegas, NV, USA. For more information see http://www.blackhat.com/.

The 21st USENIX Security Symposium will be held 8–10 August 2012 in Bellevue, WA, USA. For more information see http://usenix.org/events/.

VB2012 will take place 26–28 September 2012 in Dallas, TX, USA. The conference programme will be announced shortly. Online registration is now available. Full details can be found at http://www.virusbtn.com/conference/vb2012/.

VB2013 will take place 2–4 October 2013 in Berlin, Germany. Details will be revealed in due course at http://www.virusbtn.com/conference/vb2013/. In the meantime, please address any queries to conference@virusbtn.com.

SUBSCRIPTION RATES

Subscription price for Virus Bulletin magazine (including comparative reviews) for one year (12 issues):

- Single user: $175
- Corporate (turnover < $10 million): $500
- Corporate (turnover < $100 million): $1,000
- Corporate (turnover > $100 million): $2,000
- *Bona fide* charities and educational institutions: $175
- Public libraries and government organizations: $500

Corporate rates include a licence for intranet publication.

Subscription price for Virus Bulletin comparative reviews only for one year (6 VBSpam and 6 VB100 reviews):

- Comparative subscription: $100

See http://www.virusbtn.com/virusbulletin/subscriptions/ for subscription terms and conditions.

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire OX14 3YP, England

Tel: +44 (0)1235 555139 Fax: +44 (0)1865 543153

Email: editorial@virusbtn.com Web: http://www.virusbtn.com/