FEBRUARY 2010

virus BULLETIN

Fighting malware and spam

CONTENTS

ISSN 1749-7027

IN THIS ISSUE

ALL CHANGE

After playing the cat-and-mouse game with AV companies for several months, the author(s) of Pushdo/Cutwail finally decided to change their advanced installer, in doing so changing the communication mechanism between the servers and bots. Kyle Yang reveals the details of the new protocol.
page 4

MEMORY GAMES

Advanced memory analysis allows for rapid assessment of potentially hostile executables in memory. Ken Dunham takes us through the three phases of operation in detail: triage, capture and analysis.
page 15

VB100 ON SUSE LINUX

After the last mammoth VB100 test on Windows 7, this month sees a smaller field of competitors for the less well supported Linux platform. John Hawes reports on a mixed bag of results.
page 21

'[SSL certificate] warnings are a nuisance for the same reason they may help.'

Chester Wisniewski, Sophos

SSL CERTIFICATE WARNINGS – NUISANCE OR VALUE?

In a paper entitled 'Crying Wolf: An Empirical Study of SSL Warning Effectiveness', Carnegie Mellon researchers examined the results of a study on user behaviour in response to invalid SSL certificate warnings. While the paper makes for an interesting read on how humans perceive risk, how too much nagging breeds disregard, and how the sternness of language used to explain the situation changes perception, it mostly leads me to wonder what the point of these warnings is at all.

Primarily, the warnings protect against Man-in-the-Middle (MitM) attacks. If you are using a non-compromised browser on a non-compromised machine and manually enter your online banking URL, these warnings could alert you that something is amiss. However, I have not heard of anyone going to the effort of hijacking SSL sessions in any volume in order to execute an attack like this.

A MitM attack requires the attacker to be able to intercept/redirect the connection between a victim computer and an SSL website they are communicating with. If the interloper tried to sign a certificate convincing the client it was *Chase Bank* and they were not taking advantage of the SSL null byte vulnerability reported last summer at Black Hat, then the user would receive a warning that the certificate was self-signed or somehow modified. It's certainly possible that this would protect users, but the effort required for this type of attack tends to limit the number and scale of such attacks.

Warnings are a nuisance for the same reason they may help. Almost 100% of the time they can safely be ignored. If you receive a warning at your online banking site, more often than not the warning is because your bank's certificate has lapsed and they have been unable to retrieve a new one. This is not something your end-users should be concerned about, nor is it something they are likely to figure out based on the way browsers present the information.

Why warn your users of something that is not really indicative of an attack? This simply leads to the 'boy-who-cried-wolf' scenario. Every false warning or pop-up undermines all the work you have done as an administrator to secure a user's workspace. The users begin to second guess your entire approach and question other technologies you have used, and it can have consequences outside of the specific application.

Most attacks are phishing attacks and do not utilize SSL to begin with. If the attacker wants to be sure the padlock appears in the browser they could simply purchase a certificate for their bogus domain – it is trivial to request an SSL certificate from many providers. On 22 January I looked at approximately 100 phishes in our spam queues, and none of them used an https connection. Many URLs were obviously bogus, while some tried to manipulate the way the browser displayed the domain name to trick the user.

It is my opinion that more effective SSL warnings will not improve Internet browsing safety. The Carnegie Mellon paper does make an excellent point that may be better used in other ways. When designing software and choosing which options we present to users, we should be careful of how and when we interrupt users' work flow.

There is value in telling a user that you have blocked access to his USB drive, or that an email was blocked because multiple credit card numbers were contained in an attachment, but when you start sending alarming messages to users that they either misunderstand or that present them with a problem they cannot fix themselves, you start to create a sense that warning messages should be ignored. Users want to get back to performing the task that was interrupted and we need to design our systems to ensure that we only interrupt them when it is truly important.

Security needs to be simple and transparent to be truly effective. As an industry we need to rethink how we approach the concept of SSL certificates for verifying identity. We cannot depend on end-users to understand the implications of public key infrastructure and digital signatures and make the right choice.

NEWS

RESEARCHERS PAID FOR CHROME BUGS

Google is inviting security researchers to seek out bugs in its *Chrome* web browser. Researchers reporting genuine vulnerabilities to the company will be rewarded with $500 per bug, with the reward rising to a maximum of $1,337 if the vulnerability reported is deemed by a panel to be particularly critical. The company hopes that by encouraging external researchers to probe the browser it will be able to improve security for its users. 'The more people involved in scrutinizing *Chromium*'s code and behaviour, the more secure our millions of users will be,' read a posting on the company's blog.

The idea of offering rewards for vulnerabilities is not new, and in its announcement of the programme *Google*'s *Chromium* team raised a hat tip to the *Mozilla Foundation* whose Security Bug Bounty Program offers a similar set of rewards.

Bugs in the *Chrome* and *Chromium* builds of the browser can be reported via an online bug-tracking system. Full details can be found in the *Google* blog at http://blog.chromium.org/2010/01/encouraging-more-chromium-security.html.

PHISHING AND SCAMS CONTINUE TO RISE

According to *RSA*'s online fraud report for January, the number of phishing attacks identified in December was three per cent higher than in the previous month. *RSA* reported that, despite an apparent increase in user awareness – with 76% of respondents saying that they are familiar with phishing – the number of users that have fallen victim to a phishing attack is also on the rise. In the company's 2007 survey, five per cent of respondents admitted that they had been a victim of a phishing attack, whereas in 2009, that figure had risen to 29%. Meanwhile, research and investigation company *Ultrascan* claims that 2009 saw the highest ever annual losses from 419 fraud. The company – which collects its figures based on cases it has analysed – estimates that victims around the world lost US$9.3 billion last year – a rise from $6.3 billion in 2008.

These reports come as in the UK the Office of Fair Trading (OFT) is set to launch a scam awareness month, dubbed 'Scamnesty 2010', aiming to provide consumers with information and advice on how to identify scams and how to protect themselves. The campaign, which will run throughout February, will encourage consumers to deposit suspected scam letters in special amnesty bins in public places around the country, and to forward email scams to an 'email bin'. The campaign website (http://www.consumerdirect.gov.uk/scamnesty/) also provides examples of scam websites and advises consumers as to how to identify such sites as fake.

Prevalence Table – December 2009[1]

Malware	Type	%
Autorun	Worm	8.91%
Adware-misc	Adware	8.15%
Conficker/Downadup	Worm	7.89%
VB	Worm	6.31%
OnlineGames	Trojan	4.51%
FakeAlert/Renos	Rogue AV	4.21%
Agent	Trojan	4.17%
Heuristic/generic	Virus/worm	3.77%
Inject	Trojan	3.74%
Delf	Trojan	3.71%
Virtumonde/Vundo	Trojan	3.47%
WinWebSec	Rogue AV	3.03%
Istbar/Swizzor	Trojan	2.83%
Virut	Virus	2.81%
Alureon	Trojan	2.51%
Crypt	Trojan	1.94%
HackTool	PU	1.88%
Small	Trojan	1.86%
Zbot	Trojan	1.81%
Hupigon	Trojan	1.81%
Downloader-misc	Trojan	1.70%
Heuristic/generic	Trojan	1.10%
Bifrose/Pakes	Trojan	0.89%
Tanatos	Worm	0.89%
Peerfrag/Palevo	Worm	0.88%
Sality	Virus	0.88%
Crack	PU	0.85%
Wimad	Trojan	0.80%
Themida	Packer	0.70%
PCClient	Trojan	0.60%
Wintrim	Trojan	0.60%
BHO/Toolbar-misc	Adware	0.58%
Others[2]		10.61%
Total		100.00%

[1] This month's prevalence figures are compiled from desktop-level detections.

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

MALWARE ANALYSIS

CUT THE CUTWAIL

Kyle Yang
Fortinet, USA

After playing the cat-and-mouse game with AV companies for several months, the author(s) of Pushdo/Cutwail finally decided to change their advanced installer – previously codenamed 'Siberia' (as per the project name extracted from debugging strings in the binary) – to a brand new piece codenamed 'Revolution'. The changes mainly concern the communication mechanism between the servers and bots. More specifically, the mechanism changed from a direct communication mode using the HTTP GET method (single packet request) to a sophisticated two-way communication mode between servers and bots. In this paper, we will reveal the details of this new protocol.

WHEN AND WHY IT CHANGED

As far as we know, the Cutwail spam botnet used to seed its own executable, which usually came without an obfuscation layer. Then, around the end of October 2009, Cutwail began to seed Pushdo binaries via the infamous DHL/UPS email seeding campaign. This time it came with an obfuscation layer in order to dodge AV detection. Among the binaries was a new version string, codenamed 'Revolution6' – possibly indicating six major updates. Previously, this number only reached two ('Siberia2') – which suggests that the author(s) may have spent more time on this major update. On reversing its binary, we found that the author(s) had put a tremendous amount of work into the communication protocol, which is now both more complex and robust, and of course stealthier than the previous one (thanks to a custom packer made from a custom stub and UPX). As for what drove this change, it must certainly have something to do with the fact that the Pushdo binary and the previous communication protocol had been around for a long time, and were well known by AV companies. For instance, its critical communication packets were blocked by IPS systems. This would have disrupted the attackers' business model on a large scale, preventing them from making easy, dirty money.

Looking at our statistics, we can see that Pushdo very rapidly became one of the most prevalent threats running from October 2007 to March 2008. It is interesting to note that the original HTTP GET communication was slightly modified in December 2007 [1] to avoid detection, but seemed to quickly be thwarted as activity became significantly reduced from March 2008 onwards.

Until recently, Pushdo activity failed to return to the levels experienced during its initial outbreak. In October

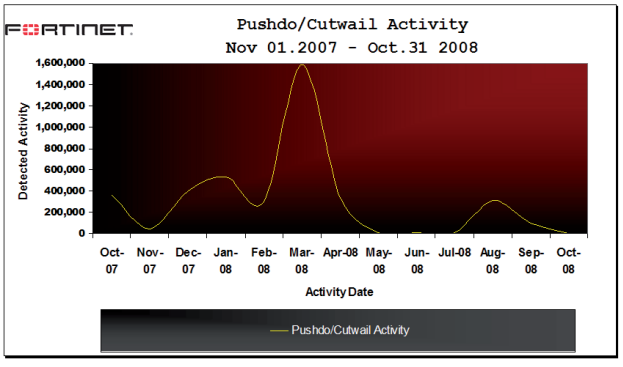


Figure 1: Initial Pushdo seeding (HTTP GET).

2009 we observed tremendous activity with the updated version: nearly ten times the level detected two years ago. This signifies an aggressive approach to seed a new, revolutionized version of the Pushdo malware installation system. Indeed, the gang behind Pushdo must have quite some resources at their fingertips with the ability to seed in such high volume.

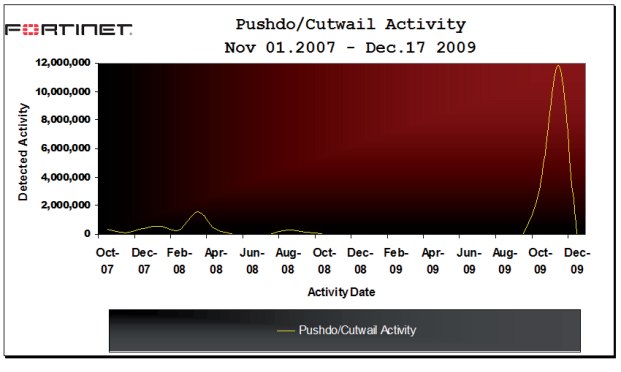


Figure 2: Pushdo/Cutwail – first gen vs. next gen.

The Revolution6 version is marked by debug ASCII strings in the unpacked binary, showing a 'pdb' filepath. PDB (program database) files contain debug and project state information for builds. In this case, the filepath shows the project name and debug name, for instance: 'f:\\programs\\revolution6\\loader\\objfre_wxp_x86\\i386\\Loader.pdb'. In some samples, while the project name remained the same, the volume changed (e.g. 'c:\\programs'), indicating that there was more than one build.

WHAT CHANGED

There are two main changes in the Pushdo payload. Firstly, in the main routine there is now a condition that leads the malware to delete itself and exit when the OS language is Russian. This is not uncommon: for instance, Conficker

avoids infecting Ukrainian systems, and various Scareware flavours do the same. In fact, there might be some connection between the Conficker and Pushdo/Cutwail gangs, although the nature of that connection remains a mystery.

The other radical change concerns the 'injection' code, the main function of which is to control the communications between the servers and the bots, for the purpose of retrieving all the malware modules (rootkit component, mailer component, etc.) that characterize the Cutwail botnet. Let's take a closer look at this process.

Nowadays, the communication channels of most infamous botnets are encrypted, either using a public encryption algorithm (e.g. AES-CBC, Base64 in Waledac, etc.) or a custom, private one. Pushdo is no exception; it resorts to many custom encryption algorithms, leveraging two main encryption/decryption keys stored in memory. Figure 3 shows the stored key pattern.

Figure 3: Encryption/decryption key.

The data circled in red is the key for encrypting/decrypting the Pushdo communication packet 'header' (made up of a start identifier and content length – eight bytes total). It should be noted that this key is hard coded in the Pushdo binary. Even more interesting is the fact that, based on the samples retrieved from different sources/spreading methods, this hard-coded key has never changed across samples. This explains why Pushdo doesn't have a routine to communicate the key to the server – the server already knows it. This is quick and efficient, and saves the cost of implementing a 'safe' exchange of symmetric keys via public key encryption schemes. But of course it is tremendously lacking in flexibility, putting the botnet at risk once the hard-coded key has been discovered (although, as we will see later, this only concerns the packet headers).

The data circled in blue is the return value of a call to QueryPerformanceCount, an API function frequently used to generate pseudo-random values. This data will be used as the key for encrypting/decrypting the Pushdo communication packet 'content'. Being generated randomly rather than pre-shared, the key must be sent to the server (see Figure 4 below).

1. Sending random encryption/decryption key to the server

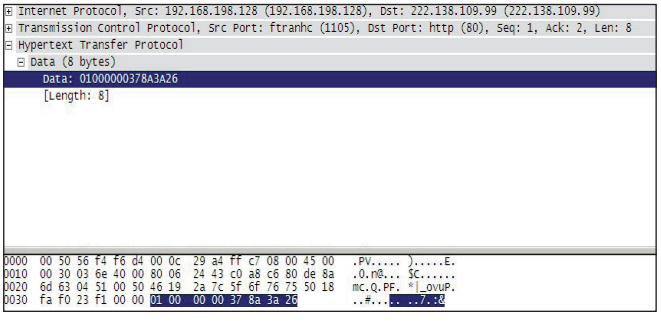


Figure 4: Sending encryption/decryption key to the server.

As can be seen above, the random key is sent to the server in plain text.

2. Sending bot and loader info to the server

The data shown in Figure 5 below is the 'clear-text' data block generated by Pushdo, which we 'intercepted' in memory prior to encryption. It appears that every data piece starts with the tag 'PROP'. This looks familiar, and is similar to the XML format used by Waledac. The difference is that Pushdo has dropped the '<tag></tag>' pairs in order to reduce the packet length and make the communication faster. Moreover, Pushdo will only send data containing the fields 'ldrver' and 'winver' to the first server it connects to (i.e. even if the rest of the communication fails, thus warranting an attempt to connect to another server, these will not be re-sent).

Let's break the whole data block into 'fields' for better comprehension. The two first fields are the 'header',

```
00083660  50 52 4F 50 1B 00 00 00 16 00 05 00 08 00 75 6E  PROP▫...▫.▫.▫.un
00083670  69 71 00 C1 84 5D 78 B7 98 9D 24 50 52 4F 50 18  iq.羹]x窒?PROP▫
00083680  00 00 00 14 00 06 00 04 00 63 6F 75 6E 74 00 00  ...▫.▫.▫.count..
00083690  00 00 00 50 52 4F 50 1A 00 00 00 14 00 08 00 04  ...PROP▫...▫.▫.
000836A0  00 73 65 73 73 69 6F 6E 00 E8 F1 CE EA 50 52 4F  .session.桉侮PRO
000836B0  50 19 00 00 00 14 00 07 00 04 00 76 65 6E 64 6F  P▫...▫.▫.▫.vendo
000836C0  72 00 11 00 00 00 50 52 4F 50 18 00 00 00 12 00  r.▫...PROP▫...▫.
000836D0  08 00 02 00 6C 64 72 74 79 70 65 00 01 00 50 52  ▫.▫.ldrtype.▫.PR
000836E0  4F 50 19 00 00 00 14 00 07 00 04 00 6C 64 72 76  OP▫...▫.▫.▫.ldrv
000836F0  65 72 00 37 00 00 00 50 52 4F 50 28 00 00 00 19  er.7...PROP(...▫
00083700  00 07 00 13 00 77 69 6E 76 65 72 00 05 00 00 00  .▫.▫.winver.▫...
00083710  01 00 00 00 28 0A 00 00 03 00 00 00 00 01 01 52  ▫...(...▫.▫...▫▫R
00083720  45 43 56 08 00 00 00                              ECV▫...
```

Figure 5: Clear data block sent from UPS/DHL spam sample.

encrypted by the pre-shared key, while the rest is the 'content', encrypted by the random key.

There are two main pieces of information in this data block. One is the basic host-related information, including volume information, system bios date, video bios data, processor model, OS ID and *Windows* version.

The other is the (malware) loader information, including loader type, loader version and vendor number. The latter is very interesting, as we are about to see. First, note that this data block is generated from a sample which is the attachment of a fake UPS/DHL email. Now, let's compare it with data from other samples with different spreading methods to see what changes in the information reported to the server by the loader.

```
00000000  50 52 4F 50 1B 00 00 00  16 00 05 00 08 00 75 6E  PROP.........un
00000010  69 71 00 37 08 64 E0 4E  0C 13 84 50 52 4F 50 18  iq.7.dàN..„PROP.
00000020  00 00 00 14 00 06 00 04  00 63 6F 75 6E 74 00 00  .........count..
00000030  00 00 00 50 52 4F 50 1A  00 00 00 14 00 08 00 04  ...PROP.........
00000040  00 73 65 73 73 69 6F 6E  00 76 D2 2A 59 50 52 4F  .session.vÒ*YPRO
00000050  50 19 00 00 00 14 00 07  00 04 00 76 65 6E 64 6F  P..........vendo
00000060  72 00 10 00 00 00 50 52  4F 50 18 00 00 00 12 00  r.....PROP......
00000070  08 00 02 00 6C 64 72 74  79 70 65 00 01 00 50 52  ....ldrtype...PR
00000080  4F 50 19 00 00 00 14 00  07 00 04 00 6C 64 72 76  OP.........ldrv
00000090  65 72 00 37 00 00 00 50  52 4F 50 28 00 00 00 19  er.7...PROP(....
000000A0  00 07 00 13 00 77 69 6E  76 65 72 00 05 00 00 00  .....winver.....
000000B0  01 00 00 00 28 0A 00 00  03 00 00 00 00 01 01 52  ....(..........R
000000C0  45 43 56 08 00 00 00 00  00 00 00 00 00 00 00 00  ECV.............
```

Figure 6: Clear data block sent from Bredolab downloaded Pushdo.

Figure 6 shows the data sent to the server by a loader seeded by Bredolab. The code shown in red highlights an

Start identifier (dd)	Data length (dd)	Key word identifier (dw)	Key word length (dw)	Content length (dw)	Key word (00 ending)	Content	Description
50 52 4F 50 PROP	1B	16	5	8	75 6E 69 71 00 uniq	C1 84 5D 78 B7 98 9D 24	encrypted data of system info
50 52 4F 50 PROP	18	14	6	4	63 6F 75 6E 74 00 count	00 00 00 00	send packet counter
50 52 4F 50 PROP	1A	14	8	4	73 65 73 73 69 6F 6E 00 session	E8 F1 CE EA	random number to identify the connection
50 52 4F 50 PROP	19	14	7	4	76 65 6E 64 6F 72 00 vendor	11 00 00 00	hard-coded in Pushdo
50 52 4F 50 PROP	18	12	8	2	6C 64 72 74 79 70 65 00 ldrtype	01 00	hard-coded in Pushdo
50 52 4F 50 PROP	19	14	7	4	6C 64 72 76 65 72 00 ldrver	37 00 00 00	hard-coded in Pushdo
50 52 4F 50 PROP	28	19	7	13	77 69 6E 76 65 72 00 winver	05 00 00 00 01 00 00 00 28 0A 00 00 03 00 00 00 00 01 01	windows version Info
52 45 43 56 RECV	08	N/A	N/A	N/A	N/A	N/A	end of the packet content

Table 1: Clear data block structure and description.

interesting difference: the vendor number is 0x10 in this sample (rather than 0x11 in our UPS sample). So what is this vendor number?

Based on the semantics of 'vendor', and on the observations above, it could be some sort of affiliate ID, used to remunerate partners who distribute the malware. In any case, it also has a practical purpose here: it indicates which method to use to get and inject the corresponding rootkit and mailer modules.

Specifically, vendor 0x11 determines that they are loaded from multiple TCP streams and a report is sent back to the server. Meanwhile, under vendor 0x10 they are loaded in a single TCP stream and no report is sent back to the server. The fact that this vendor number is hard coded in the Pushdo binary indicates that the author(s) made independent/custom/private binaries for different 'vendors'; in the process, they introduced a new operation mode – rush mode (more on which later).

In order to better reveal the different server responses which could be possible based on the 'vendor', we carried out some fuzzing on the Pushdo request packet (more details later).

After generating this data buffer for the server, Pushdo will encrypt it prior to sending it. As we have hinted before, the encryption routine has two separate phases. One uses the hard-coded key to encrypt the start identifier and data length ('header'). The other uses the randomly generated key that has already been sent to the server to encrypt the rest of the data ('content').

Encryption algorithm A: encrypt start identifier and data length

In this case, the 'Start Identifier' and 'Data Length' are simply XORed with the hard-coded key.

For example:

```
50 52 4F 50 1B 00 00 00 ^ 45 9A B3 61   8E 20 3F 19 =
15 C8 FC 31   95 20 3F 19

52 45 43 56 08 00 00 00 ^ 45 9A B3 61   8E 20 3F 19 =
17 DF F0 37   86 20 3F 19
```

As mentioned previously, the key is likely pre-shared with the server (albeit the server could get this encryption key simply by XORing the original data (50524F501B0000) with the encrypted data received from the bot).

Encryption algorithm B: encrypt rest of data

In this case, the first three bytes of the randomly generated key are used. For each byte of data to encrypt, the remainder of the quotient 'byte offset / 4' is computed, and its value is the index of the encryption operation to apply to that byte.

For example:

Key: 37 8A 3A

Offset % 4	Method
0	Byte ^ 37
1	Byte - 8A
2	Byte + 3A
3	Byte ^ (3A + 8A)

Table 2: Data encryption algorithm.

Original data: 16 00 05 00 08 00 75 6E

Encrypted data: 21 76 3f c4 3f 76 AF AA

The encrypted block is then sent to the server. The interesting thing is that the server could use the Volume Info in the 'uniq' to identify whether the bot is running under a virtual machine environment, but it seems to ignore this.

3. Retrieving other components

There are two possible kinds of reply from the server, which are identified by their 'start identifier'. Those with 'PROP' as a start identifier contain the server list. Pushdo will update its list based on this message. Those with 'FILE' as a start identifier contain other components, including rootkits and the spam engine. As mentioned before, Pushdo has two operation modes for retrieving those (based on the vendor number): stepping mode and rush mode.

Stepping mode

Just as its name implies, in stepping mode Pushdo will retrieve the rootkit and spam engine step by step, one by one. More specifically, it will send the bot status back to the server to determine the next step after every module/list retrieved (see Figures 7–9).

Table 3 breaks down the whole data block into fields.

The retrieved message process routine depends on the 'TMP file determiner' and 'file type identifier'. If the 'file type identifier' equals 1, then it will check the 'TMP file determiner'. If a result of 1 occurs after a logical AND, a temporary file will be created before injecting into a suspend mode svchost.exe. Otherwise, it will be injected directly into a suspend mode svchost.exe, but it will always perform a sanity check before the injection. The svchost.exe process info will be stored in the memory and a partial file checksum value will be used when reporting bot status. Figure 10 shows the structure of the bot status info.

Type	Start identifier (dd)	Data length (dd)	TMP file determiner (dw)	Key word length (dw)	File type identifier (dw)	File length (dd)	Content identifier (dd)	Key word (00 Ending)	Content
Rootkit installer	46 49 4C 45 FILE	0x9224	2	0B	1	0x9200	0x34	52 6F 6F 74 4B 69 74 5F 31 37 00 00 RootKit_17	N/A
Spam engine	46 49 4C 45 FILE	0x45E2A	0	11	1	0x45E00	0x12	4D 61 69 6C 65 72 5F 52 53 31 5F 65 6D 70 74 79 00 00 Mailer_RS1_empty	N/A
Relay server info	46 49 4C 45 FILE	0x5E	0A	22	20	0x4	0x63	4D 61 69 6C 65 72 5F 52 53 31 5F 37 38 5F 31 35 39 5F 31 32 31 5F 34 31 28 44 69 6D 61 72 69 6B 29 00 Mailer_RS1_78_159_121_41(Dimarik)	/host:78.159.121.41 /port:38811

Table 3: FILE message structure.

```
00000020  46 49 4C 45 24 92 00 00 02 00 00 00 0B 00 01 00  FILE$?.▫... .▫.
000003000 92 00 00 34 00 00 00 52 6F 6F 74 4B 69 74 5F     .?.4...RootKit_
00000040  31 37 00 00 4D 5A 90 00 03 00 00 00 04 00 00 00  17..MZ?▫...▫...
00000050  FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00  ▫▫..?......@...
00000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000080  D8 00 00 00 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C  ?..?.???L
00000090  CD 21 54 68 69 73 20 70 72 6F 67 72 61 6D 20 63  ?This program c
```

Figure 7: Clear FILE data block – rootkit installer (partial).

```
00000020  46 49 4C 45 2A 5E 04 00 00 00 00 00 11 00 01 00  FILE*^..........
00000030  00 5E 04 00 12 00 00 00 4D 61 69 6C 65 72 5F 52  .^......Mailer_R
00000040  53 31 5F 65 6D 70 74 79 00 00 4D 5A 90 00 03 00  00S1_empty..MZ....
00000050  00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00  ......ÿÿ..¸.....
00000060  00 00 40 00 00 00 00 00 00 00 00 00 00 00 00 00  ..@.............
00000070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000080  00 00 00 00 00 00 F0 00 00 00 0E 1F BA 0E 00 B4  ......ð.....°..´
00000090  09 CD 21 B8 01 4C CD 21 54 68 69 73 20 70 72 6F  .Í!¸.LÍ!This pro
```

Figure 8: Clear FILE data block – spam engine (partial).

```
00000020  46 49 4C 45 5E 00 00 00 0A 00 00 00 22 00 20 00  FILE^.......". .
00000030  04 00 00 00 63 00 00 00 4D 61 69 6C 65 72 5F 52  ....c...Mailer_R
00000040  53 31 5F 37 38 5F 31 35 39 5F 31 32 31 5F 34 31  S1_78_159_121_41
00000050  28 44 69 6D 61 72 69 6B  29 00 2F 68 6F 73 74 3A  (Dimarik)./host:
00000060  37 38 2E 31 35 39 2E 31  32 31 2E 34 31 20 2F 70  78.159.121.41 /p
00000070  6F 72 74 3A 33 38 38 31  31 00 12 00 00 00 7E D6  ort:38811.....~Ö
```

Figure 9: Clear FILE data block – mail relay server.

```
00000020 00 01 00 48 CB 0A 00 00 92 00 00 00 00 00 00 02
00000030 10 00 00 34 00 00 00
```

Descriptions:
00 01 - Running Injected Process Counter
48 CB 0A 00 - Process Address
00 92 00 00 - File Length
02 10 00 00 - Partial File CheckSum Value
34 00 00 00 - Content Identifier

Figure 10: Bot status data – one process (rootkit installer) running.

Pushdo will use the bot status data to form the report communication packet. Figure 11 shows an example of the content of such a report packet.

If the injection fails, or if the bot needs to change to another server (perhaps because the current connection broke), Pushdo will always retrieve the rootkit installer until the injection succeeds. Therefore, the 'botstatus' could contain information for more than one running process. This also indicates that Pushdo has the ability to load other malware and report back to the server.

Rush mode

The major difference between rush mode and stepping mode is the absence of reports ('botstatus') in rush mode and the fact that the rootkit installer and spam engine are retrieved at the same time (single TCP stream). Thanks to the well-organized data structure, Pushdo can process them as easily as in stepping mode and even faster. Bredolab has been downloading Pushdo binaries, and this mode has only been seen in the Bredolab-downloaded Pushdos. The report will only be sent when the process is running. It's a possibility that Bredolab just tries to use the Cutwail spam engine to spread itself.

```
00000000                                  50 52 4F 50 1B 00 00 00   PROP▫...
00000010  16 00 05 00 08 00 75 6E 69 71 00 45 19 CB F8 37   ▫.▫.▫.uniq.E▫▪7
00000020  52 55 32 50 52 4F 50 18 00 00 00 14 00 06 00 04   RU2PROP▫...▫.▫.▫
00000030  00 63 6F 75 6E 74 00 03 00 00 00 50 52 4F 50 1A   .count.▫...PROP▫
00000040  00 00 00 14 00 08 00 04 00 73 65 73 73 69 6F 6E   ...▫.▫.▫.session
00000050  00 BA F9 DF 5C 50 52 4F 50 19 00 00 00 14 00 07   .葫運PROP▫...▫. ▫
00000060  00 04 00 76 65 6E 64 6F 72 00 11 00 00 00 50 52   .▫.vendor.▫...PR
00000070  4F 50 18 00 00 00 12 00 08 00 02 00 6C 64 72 74   OP▫...▫.▫.▫.ldrt
00000080  79 70 65 00 01 00 50 52 4F 50 22 00 00 00 19 00   ype.▫.PROP"...▫.
00000090  0A 00 0A 00 62 6F 74 73 74 61 74 75 73 00 34 00   ....botstatus.4.
000000A0  00 00 00 00 02 10 00 00 52 45 43 56 08 00 00 00   ....▫ ▫..RECV▫...
```

Figure 11: Clear report data block.

FUZZING PUSHDO BOTNET REQUEST PACKET

The main purpose of this part of the paper is to reveal the relationship between different vendor ID and server responses, and to gain an idea of how many custom/private Pushdo binaries there are now, by fuzzing the vendor value.

Method: We use protocol re-writer to customize the vendor value before it sends out, then intercept the response packets.

The vendor ID with corresponding server response will be the following (partial):

Rootkit_[number] represents the 'Key Word'.
Loader_V0x[number] represents the Pushdo binary with hard-coded vendor 'number'

Vendor=0x00, downloaded Modules:
Rootkit_1, Loader_V0x01, Mail Sniff, Filter

Vendor=0x01, downloaded Modules:
Rootkit_1, Loader_V0x1, Mail Sniff, Filter

Vendor=0x02, downloaded Modules:
Rootkit_2, Loader_V0x2

Vendor=0x09, download Modules:
Rootkit_9, Loader_V0x9

Vendor=0x10, downloaded Modules:
Rootkit_16, Spam Engine
Spam Engine served for Bredolab.

Vendor=0x11, downloaded Modules:
Rootkit_17, Spam Engine
Spam Engine served for Pushdo itself.

Vendor=0x12, downloaded Modules:
Rootkit_18, Loader_V0x12, Spam Engine
Spam Engine served for ???

Vendor=0x13, downloaded Modules:
Rootkit_19, Loader_V0x13,
Spam Engine
Spam Engine served for ???

Vendor=0x14, downloaded Modules:
Rootkit_20, Loader_V0x14,
Spam Engine
Spam Engine served for ???

Vendor=0x15, downloaded Modules:
Rootkit_21, Loader_V0x15,
Spam Engine
Spam Engine served for ???

Vendor=0x16, downloaded Modules:
Rootkit_22, Loader_V0x16, Spam Engine, Web Mailer
Spam Engine served for ???

Vendor=0x17, downloaded Modules:
Rootkit_23, Loader_V0x17, Spam Engine
Spam Engine served for ???

Vendor=0x18, downloaded Modules:
No Modules downloaded.

Vendor=0x19, downloaded Modules:
No Modules downloaded.

Vendor=0x1A, downloaded Modules:
No Modules downloaded.

Vendor=0x1B, downloaded Modules:
Rootkit_1, Loader_V0x01, Web Mailer

Vendor=0x20, downloaded Modules:
Rootkit_1, Loader_V0x01, Web Mailer

Vendor=0x63, downloaded Modules:
Rootkit_1, Loader_V0x1

Vendor=0xfd, downloaded Modules:
Rootkit_1, Loader_V0x01

Vendor=0xfe, downloaded Modules:
Rootkit_1, Loader_V0x01, Web Mailer

Vendor=0xff, downloaded Modules:
Rootkit_1, Loader_V0x01

Vendor=0x100, downloaded Modules:
Rootkit_1, Loader_V0x01

Vendor=0xfffe, downloaded Modules:
Rootkit_1, Loader_V0x01, Web Mailer

Vendor=0xffff, downloaded Modules:
Rootkit_1, Loader_V0x01

Vendor=0xffffff, downloaded Modules:
Rootkit_1, Loader_V0x01

Vendor=0xffffffff, downloaded Modules:
Rootkit_1, Loader_V0x01

From the above fuzzing results (gathered 18 December 2009), we can see that both the rootkit and loader version number is related to the vendor ID. There may also be 23 private/custom Pushdo binaries based on the different hard-coded vendor IDs. As mentioned previously, the vendor may be related to the partners who distribute the malware. In this case we only revealed two: 0x10 and 0x11, but there are still more to be revealed. We have even seen some new modules with another new project name: 'webbot'. The debug build has now reached three, 'WebMailer3'. After reversing this binary, it appears to still be experimental and that the author(s) are still trying to debug it since there is a '/debug' switch. *Fortinet*'s *FortiGuard Labs* has dubbed this new engine 'Webwail'. Webwail has the ability to register new email accounts and send spam from the web. Around 15 January 2010, Pushdo started to download this new spam engine, spreading Webwail with the help of its old friend Bredolab once again. At the time of writing, Webwail is registering *Hotmail* accounts [2].

DETECTION

Thanks to its well-organized data structure, Pushdo can communicate a large amount of information from and to its command and control server. We can read that information as well, the only issue being the encryption key. However, the encryption key doesn't change even in custom/private builds for other malware, thus making detection easier.

REFERENCES

[1] SecureWorks. http://www.secureworks.com/research/threats/pushdo/.

[2] More information on Webwail can be found at http://blog.fortinet.com/bredolab-gearing-up-to-spam-the-web/.

[3] More information on Pushdo/Cutwail can be found at http://www.fortiguard.com/analysis/pushdoanalysis.html.

[4] More information on Crime Services can be found at http://blog.fortinet.com/adaptive-crime-services/.

CALL FOR PAPERS

VB2010 VANCOUVER

Virus Bulletin is seeking submissions from those wishing to present papers at VB2010, which will take place 29 September to 1 October 2010 at the Westin Bayshore hotel, Vancouver, Canada.

The conference will include a programme of 30-minute presentations running in two concurrent streams: Technical and Corporate.

Submissions are invited on all subjects relevant to anti-malware and anti-spam. In particular, *VB* welcomes the submission of papers that will provide delegates with ideas, advice and/or practical techniques, and encourages presentations that include practical demonstrations of techniques or new technologies.

A list of topics suggested by the attendees of VB2009 can be found at http://www.virusbtn.com/conference/vb2010/call/. However, please note that this list is not exhaustive, and the selection committee will consider papers on these and any other anti-malware and anti-spam related subjects.

SUBMITTING A PROPOSAL

The deadline for submission of proposals is **Friday 5 March 2010**. Abstracts should be submitted via our online abstract submission system. You will need to include:

- An abstract of approximately 200 words outlining the proposed paper and including five key points that you intend the paper to cover.
- Full contact details.
- An indication of whether the paper is intended for the technical or corporate stream.

The abstract submission form can be found at http://www.virusbtn.com/conference/abstracts/.

One presenter per selected paper will be offered a complimentary conference registration, while co-authors will be offered registration at a 50% reduced rate (up to a maximum of two co-authors). *VB* regrets that it is not able to assist with speakers' travel and accommodation costs.

Authors are advised that, should their paper be selected for the conference programme, they will be expected to provide a full paper for inclusion in the VB2010 Conference Proceedings as well as a 30-minute presentation at VB2010. The deadline for submission of the completed papers is Monday 7 June 2010, and potential speakers must be available to present their papers in Vancouver between 29 September and 1 October 2010.

Any queries should be addressed to editor@virusbtn.com.

FEATURE

DATA TAINTING FOR MALWARE ANALYSIS – PART THREE

Florent Marceau
CERT-LEXSI, France

In this three-part series Florent Marceau studies the use and advantages of full virtualization in the security field. Following an introduction to full virtualization (see *VB*, September 2009, p.6), and a look at the limitations of the technology (see *VB*, November 2009, p.8), the third and final part looks at the implementation of the technology.

IMPLEMENTATION

Our mechanism of data tainting as described previously will be the engine of our project. Every byte of RAM will have a tag in the taintmap. As we have seen previously, each byte received from the network (in order to characterize the configuration file) will be marked and propagated. This propagation is guaranteed in RAM as well as on the hard drive. Every malware sample we wish to analyse will be loaded from a virtual CD; therefore we will taint any data coming from the CD in order to mark the binary image to be analysed. This way, the code and the binary data of the malware are tainted at the outset. When the binary unpacks itself, the tainted code of the unpacker will read its similarly tainted data in order to generate the viral code – which will also be tainted thanks to the propagation mechanism.

1.1 Taint mark differentiation

Since we use a tag size of one byte, we can use eight different bit flags. These flags will be used to differentiate between the tainted data that originates from the network and the data loaded from the disk or the CD.

This simple scheme allows us to distinguish network data from the data that is already present in the system. However, when data is manipulated by the malware code, it may be combined with the hard drive tainted data. Which flag should persist then? We need to define a persistence policy for tags. If data received from the network is saved immediately to the disk for further processing, it must keep its network taint mark and not that of the hard drive. Meanwhile, a tainted piece of hard drive data will never acquire the network tag (indeed, for our purposes, outgoing traffic is of no interest).

1.2 Data dumping

The final part of our implementation is the dumper code. The goal here is to capture all data from the network that has been manipulated by a piece of tainted code (regardless of the code's mark of origin). This allows us to characterize an unciphered configuration file.

The dumping code must reside in a place where the data flow can easily be controlled. We implement it as MMU hooks in order to control all reading and writing exchanges between the RAM and the CPU. These hooks must be placed and executed just before the legitimate operation; in this way, in the case of a write operation, we can check the data that will be written as well as the data that will be overwritten. For performance reasons, this code must be optimized (written in inline assembly) since its execution will involve a virtual RAM access latency that will have a heavy impact.

We introduce here the use of another bit of the tag to distinguish between tainted data that has been manipulated and other, intact tainted data. We define manipulated data as any tainted data written in RAM from a piece of tainted code. The manipulated bit is not like the other flags that are used to distinguish the data's origin (network or drive). The only purpose of this bit is to trigger the dumping mechanism. When a piece of tainted data is written from a given piece of tainted code, this data will be marked as manipulated. Subsequently, any access of this manipulated data area (reading or writing) will trigger the dumping mechanism. This will dump the data area in an output file and then delete the manipulated bit (but not the origin marker). Thus, the manipulated bit is not persistent (and is not propagated) in order to minimize redundancy in our dump file. This also allows the dump size to be limited and ensures that each layer of decryption and decompression will be captured.

To summarize, in RAM we now have the original tainted data (from the network or from the disk), and among this data, some has been manipulated by tainted code and is marked as such.

Now we only have to make the dumping logic as exhaustive as possible. For this, we chose the following implementation:

- On a write access (CPU -> RAM) (an overwrite case) from a non-tainted piece of code (such as a part of the OS): if the target address in RAM is tainted and has been manipulated, we capture it in a 'net_dump' file if it consists of network data, and otherwise we capture it in an 'other_dump' file.
- On a write access (CPU -> RAM) (an overwrite case) from a tainted piece of code (such as the malicious code): if the data to be written in RAM is tainted, we add the manipulated bit to its flag. For the data to be overwritten, as previously, if the target address in RAM

is tainted and has been manipulated, we capture it in a 'net_dump' file if it consists of network data, and otherwise we capture it in an 'other_dump' file.

- On a read access (RAM -> CPU): if the data to be read in RAM is tainted and has been manipulated, we capture it in a 'net_dump' file if it consists of network data, and otherwise we capture it in an 'other_dump' file.

In order to improve the efficiency and readability of the dump, the dumping mechanism will dump the data and all other tainted data that are contiguous with it. For this purpose, the dump mechanism scans the memory from the targeted addresses to the lower addresses of the tainted data, thereby determining the lower boundary of the dump area. The upper boundary will be determined in the same way so that we know the exact dumping scope.

Figure 1 illustrates the dumping process.

1.3 Limitations of the dumper mechanism

The dumping mechanism is not perfect, and is not our ideal choice, but the mechanism has to be as effective as possible for a maximum number of samples (an empirical approach). Let's consider a decryption algorithm such as:

```
( ) lea esi, [data]
( ) mov ecx, SIZE
( ) decode:
( ) mov eax,[esi]
( ) xor eax,0xd3adc0de
( ) mov [esi],eax
(1) add esi,3
( ) loop decode
```

This simple code will decode the data with a xor opcode using the key 0xd3adc0de. The problem here lies in (1) – at each iteration the different decoded zones overlap themselves. With the previous implementation of our dumping mechanism, during the first iteration, the first three bytes of the string would be dumped in clear text, with the fourth byte not yet decoded. The fourth byte would be decoded and dumped again only at the second iteration. The results here would then be the dump of the clear text string, yet every three bytes the text would be discontinued by a garbage data byte. Other examples are possible; there is no absolute solution here, we just have to find the one that will be the most effective.

1.4 Empirical observations

This project was put into practice two years ago. Regular monitoring of valid codes clearly shows that our main obstacles are the use of particularly heavy obfuscation and new virtual machine detection techniques. If we

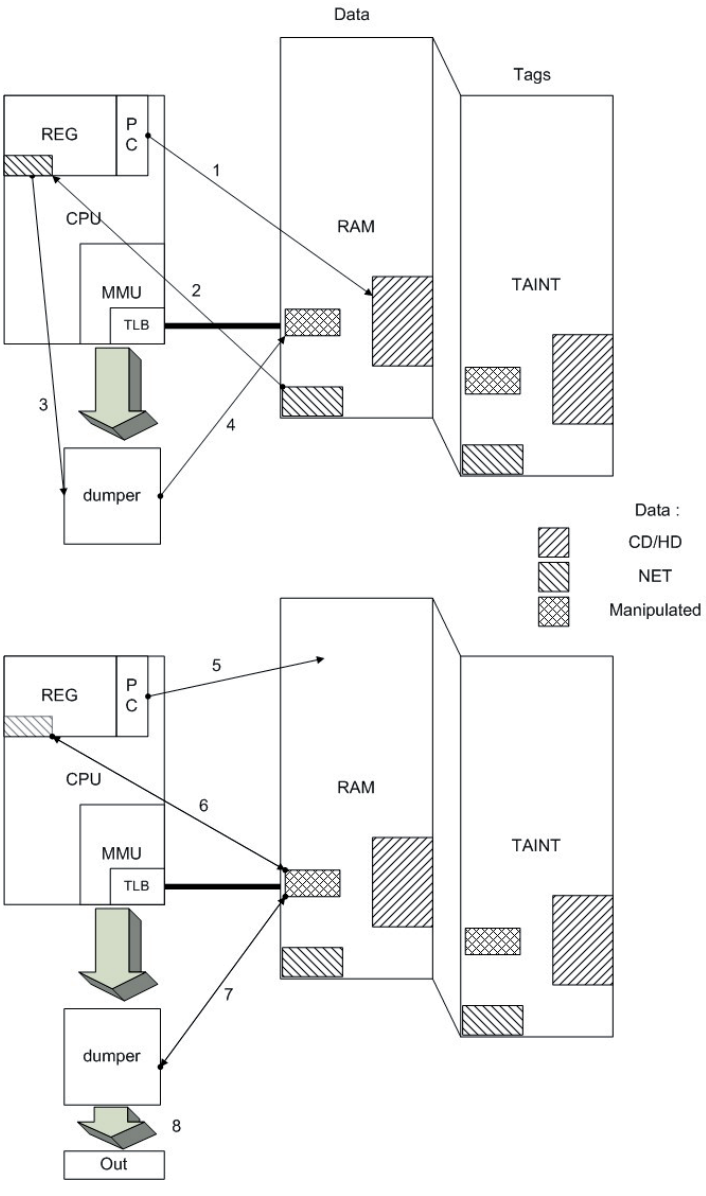


Figure 1: Within RAM we have tainted data that originates from the network or from the different volumes. An operation from a piece of tainted code (1), such as reading network data (2) in order to apply some arithmetic processing to it, leaves this data tainted on registers. When storing it in RAM, the dumper (3) adds the 'modified' mark (4). Later, whatever the origin of the operation (5), any accessing of the previous data (read / write) (6) triggers the dumper mechanism to evaluate the size of the modified tainted data (7), and then remove this modified mark to finally dump the data in its output file (8).

consider the global evolution of banker trojans we can conclude that there has been a great advancement in the malicious techniques used. Many have embedded 'kernel rootkit' capabilities, which won't impact our solution, but the use of increasingly sophisticated packers and complex encryption algorithms may become a problem. We rarely find malware with anti-tainting capabilities, although anti-VM methods have become common. Thus, an implementation of data tainting that is limited to 'explicit direct flow' propagation is currently still sufficient (although this could change if there is an increase in the number of automated analysis platforms based on data tainting).

2. RESULTS

In this section we look at the results of applying our data tainting mechanism to real-world malware.

2.1 Torpig/Sinowal family

The configuration file for this piece of malware is the same for all samples and uses a weak encryption (xor) with a constant key.

The interest here is purely technical, more than a year ago the first versions of our platform were 100% effective against this malware family. However, with the emergence of a newer variant using the MBR rootkit Mebroot/MaOS [1], the encryption applied by the rootkit (on the rootkit itself and its downloaded files) became so strong that the code propagation tended to be lost. Unfortunately (or perhaps fortunately), this variant was available for only a short period of time and was rapidly taken offline. Thus, we didn't have time to complete our tests on this sample.

A new active sample of Torpig/Mebroot was found in April, which we were able to test on our platform. Results showed that, despite the heavy encryption and the rootkit deployment stage, our implementation of the data tainting mechanism, while significantly slowed, was fully effective. The results are shown below:

MD5: d438c3cb7ab994333fe496ef04f734d0
Hard drive dump file size: 1.2G
Network dump file size: 28M

We then get the following dll configuration on the network dump file:

```
(...)
?{ba1r}pbb,?pa~eps}tJO/L:/1beh}t,gxbxsx}xeh+yxuut
66.29.115.68
66.29.115.68
kolpinik.com
mikorki.com
pibidu.com
online.westpac.com.au ib.national.com.au www1.
maxisloans.com.au
*.inetbank.net.au access.imb.com.au www.homebank.
com.au www.etradeaustralia.com.au secure.esanda.com
is2.cuviewpoint.net onlineteller.cu.com.au ebanker.
arabbank.com.au onlineserv
ices.amp.com.au
*advisernet.com.au *boq.com.au secure.accu.com.au
*citibank.com.au secure.ampbanking.com www3.netbank.
commbank.com.a *cua.com.au ibank.communityfirst.com.au
ib.bigsky.net.au online.mecu.com.au
*citibankonline.ca
service.oneaccount.com www.mybusinessbank.co.uk
*npbs.co.uk ibank.barclays.co.uk *banking.
bankofscotland.co.uk www.bankofscotlandhalifax-
online.co.uk *citibank.co.uk *icicibank.
co.uk *adambanking.com *capitalonesavings.co.uk
www*.440strand.com www.nwolb.com www.rbsdigital.com
myonlineaccounts*.abbeynational.co.uk welcome*.smile.
co.uk welcome*.co-operativebank.co.uk *natwest.co.uk
*rbsdigital.co.uk www.mybank.alliance-leicester.co.uk
ibank.cahoot.com online*.lloydstsb.* home.cbonline.
co.uk *ybonline.co.uk
*cseb*.it hbnet*.cedacri.it servizi.allianzbank.it
servizi.atime.it *cabel.it homebanking.cariparma.
it www.in-bank.net *isideonline.it www.linksimprese.
sanpaoloimi.com www.nextbanking.it *bam.it
*bancatoscana.it *mps.it
www.sparkasse.at www.banking.co.at *cortalconsors.de
bankingportal.*.de finanzportal.fiducia.de banking.*.de
portal.*.de internetbanking.gad.de *postbank.
de *apobank.de *dkb.de *haspa.de *reuschel.com
*citibank.de *hypovereinsbank.de *bulbank.bg
paylinks.cunet.org *bankatlantic.web-access.com
*businesse-cashmanager.web-access.com *suntrust.com
*cashproweb.com *ebanking-services.com netteller.com
*wamu.com *ameritrade.com
*bancopopular.com
*cbbusinessonline.com *paypal.com *ebay.com *53.com
*airforcefcu.com *aol.com *banking.firsttennessee.
biz *banking.firsttennessee.com *bankofamerica.
com *bankofbermuda.com *bankofoklahoma.com
*bankofthewest* *capitalone.com *chase.com *cib.
ibanking-services.com *citibank.com *citibusiness.com
*citizensbankonline.com
*columbiariverbank.com *comerica.com *community-
boa.com *dbs.com *dollarbank.com *firstib.com
*firsttennessee-loan* *jpmorganinvest.com *key.com
*lasallebank.com *lehmanbank.com
*military-boa.com *nationalcity.com *navyfcu.com
*ncsecu.org *ocfcu.org *onlinebank.com *onlinesefcu.
com
*peoplesbank.com *selectbenefit.com *sharebuilder.com
*site-secure.com *tcfbank.com *tcfexpressbusiness.
com *uboc.com *us.hsbc.com *usaa.com *usbank.com
*wachovia.com *wellsfargo.com
*ubs.com *raiffeisendirect.ch *postfinance.ch
*migrosbank.ch *bekb.ch *blkb.ch *netbanking.ch
```

```
*lukb.ch *zkb.ch *bank.ch *bcvs.ch *bcge.ch
banking.*.ch *vontobel.com *ubp.ch *sarasin.ch *hbl.
ch *directnet.com *arabbank.ch *baloise.ch

*alpha.gr *bankofcyprus.gr *marfinegnatiabank.gr
*winbank.gr *eurobank.gr *nbg.gr *millenniumbank.gr
*piraeusbank.com *emporiki.gr

*centralbank.gov.cy *bankofcyprus.com *laiki.
com *usb.com.cy *hellenicbank.com *coopbank.com.cy
*universalbank.com.cy

*uno-e.com www*.bancopopular.es www.bv-
i.bancodevalencia.es oi.cajamadrid.es

net.kutxa.net telemarch.bancamarch.
es *bancocaixageral.es *caixagirona.es www.
caixacatalunya.es *bbva.es *bbvanetoffice.com
telemarch.bancamarch.es

bancae.bancoetcheverria.es lo*.lacaixa.es www.
cajacanarias.es areasegura.banif.es seguro.cam.
es www.fibanc.es *sanostra.es www.inversis.com oie.
cajamadridempresas.es

vs*.absa.co.za mijn.postbank.nl

marsco.com vmd.ca Citrix scottrade.com streetscape.
com principal.com thinkorswim.com sharebuilder.com
fs.ml.com netxselect.com netxclient.com

accu.com.au adelaidebank.com.au amp.com.au bigsky.
net.au boq.com.au commbank.com.au communityfirst.com.
au cu.com.au cua.com.au imb.com.au inetbank.net.au

mecu.com.au nab.com.au suncorp.com.au westpac.com.au
hsbc.com.au bankwest.com.au bendigobank.com.au necu.
com.au comsec.com.au ebanking.pcu.com.au

teacherscreditunion.com.au policecredit.com.au/
easyaccess stgeorge.com.au banksa.com.au humebuild.
com.au zecco.com etrade tradingdirect.com ameriprise.
com

businesscreditcardsonline.co.uk alpha.gr
bankofcyprus.gr marfinegnatiabank.gr winbank.gr
eurobank.gr nbg.gr millenniumbank.gr piraeusbank.com
emporiki.gr

centralbank.gov.cy bankofcyprus.com laiki.com usb.
com.cy hellenic coopbank.com.cy universalbank.com.cy

anbusiness.com paypal.com hellenicbank
citibankonline.ca clkccm cashplus capitalonebank.com
nationalcity.com webcashmanager cashman towernet

web-access.com cashproweb.com bankonline.sboff.
com constantcontact.com/login.jsp dotmailer.co.uk/
login.aspx yourmailinglistprovider.com/controlpanel
r57shell.php c99shell webadmi

(...)
```

And the heavily ciphered rootkit configurations files:

```
(...)

INST

gc00

services.exe

!This program cannot be run in DOS mode.

(...)

0$0/050;0D0J0Q0W0^0d0i0

gs00

!winlogon.exe;services.exe;csrss.exe;spoolsv.exe;
lsass.exe;smss.exe;system.exe

!This program cannot be run in DOS mode.

(...)
```

We were therefore able to verify that the taint propagation on the hard drive reflects the MBR contamination and also the contamination of the following sectors containing the rootkit itself.

The previous check was only done in order to validate the tainting at the hard disk level in the presence of an MBR-infecting virus. The original version of Mebroot [1] uses sectors 60/61/62 to store its boot loader and the original boot sector, but contrary to our expectations, these sectors were not tainted. Indeed, after a quick analysis, the new variant sample (d438c3cb7ab994333fe496ef04f734d0) had partially changed its modus operandi – its boot loader and the original boot sector were now placed at the end of the hard disk. This new variant is immune to any detection/disinfection using the anti-rootkit gmer (and mbr.exe [2]). This is probably also the case for many of today's anti-virus solutions.

2.2 Further results

Similar sets of results for the PRG/Zeus/NTOS, Infostealer, Ambler, Banker and PWS-OnlineGames.cz families can be found at http://www.virusbtn.com/vba/2010/02/vb201002-data-tainting-results.

CONCLUSION

In this paper we have revisited many concepts of the old binary instrumentation domain. In the early 90s *DEC* applied this kind of technology to translate VAX binary images to alpha platforms; in the 70s *IBM* applied it to provide debugging.

Nowadays, superior hardware performance allows virtualization and binary analysis to be used in new ways. We have shown here one such application, which is usable on real-life malicious software. In the future we plan to improve our implementation by adding static binary analysis and constraint solving capabilities.

REFERENCES

[1] Florio, E.; Kasslin, K. Your Computer is Now Stoned (...Again!) The Rise of MBR Rootkits. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/your_computer_is_now_stoned.pdf.

[2] Gmer: Stealth MBR rootkit. http://www2.gmer.net/mbr/.

TUTORIAL

INTRODUCTION TO ADVANCED MEMORY ANALYSIS

Ken Dunham
iSIGHT Partners, USA

Cybercriminals are pushing fraud to the limits, now resorting to memory-only tactics to subvert the *Windows* operating system for financial gain. These recent tactics pose a challenge for traditional forensics, law enforcement, auditing and incident response procedures, and require new ways of dealing with affected systems. Advanced memory analysis allows for rapid assessment of potentially hostile executables in memory. There are three distinct phases of operation: analysis of a live system (triage), dumping of volatile data to a file (capture), and analysis of combined data (analysis).

TRIAGE

Incident handlers often locate infected computers through egress traffic via IDS/IPS-type solutions. Once a computer is known to be sending suspect or known hostile traffic, the hunt begins to find the offending process, image files, and scope of compromise. While some malicious programs hide data in Alternate Data Streams (ADS) and other tricky places, a movement towards kernel-level rootkit subversion and RAM-only code is underway in the wild. As a result, incident handlers who are not equipped with advanced stealth code identification tools and techniques are unable to triage a system properly to identify potential kernel-level rootkits on an infected host.

Triage begins with the age-old basics of incident handling and forensic techniques. Ideally, policy enables the incident handler to collect information relating to volatile data on the system. If policy only allows a hard-core traditional forensic approach, critical volatile RAM-only data will be lost. To focus on memory analysis and volatile data, incident handlers must begin triage with operations such as the following, which are largely focused on malicious process and image identification:

1. Windows Task Explorer (CTRL-ALT-DELETE)

a. If the *Task Explorer* won't open, this is a sign that malicious code may be hindering the use of security tools on the system.

b. Once the *Task Explorer* is open, look for anything in the list of processes that shouldn't be there. While it is increasingly rare, malicious programs do sometimes still reveal themselves in *Windows Task Explorer*. Even when something is visible in memory, other components of an attack may be hidden.

c. Also look for anything that is missing from the list of processes. For example, one variant of Haxdoor injects explorer.exe and then hides the process it injects. Explorer.exe should always be visible within *Windows Task Explorer*; if it isn't, a rootkit is probably hiding it.

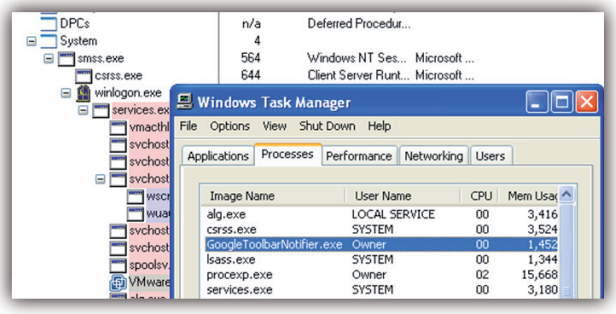


Figure 1: Explorer.exe is hidden by a rootkit and isn't visible when it should be.

2. Process Explorer (http://technet.microsoft.com/en-us/sysinternals/bb896653.aspx)

Following the same procedures as when working with *Windows Task Manager*, look for extra and/or missing processes such as explorer.exe, csrss.exe and lsass.exe. Although it is increasingly unlikely that any rootkit processes will be found using this method, it is worthy of investigation since some malicious programs are visible using this tool but not with *Windows Task Manager*, indicating a possible rootkit process.

3. F-Port (http://www.foundstone.com/us/resources/proddesc/fport.htm)

Run *F-Port* and dump the results to a file. It is a good idea to have a baseline dump from a known clean system to compare against the dump from the possibly infected system. This enables the incident handler to identify what *F-Port* has seen in memory mapped to specific ports and file images at the time of analysis, and to quickly identify what might be malicious.

Looking at the dumps shown in Figure 2, can you identify which one is from an infected system? Column B is longer for a reason: several new processes have been spawned by explorer.exe. This suggests an injected malicious process attempting to communicate with a remote C&C server.

By performing a standard 'diff' or difference analysis (common in the forensics field), an incident handler can very quickly identify potentially hostile processes. To do this even more quickly, consider using the *FCompare* utility

Dump A

```
Pid   Process             Port  Proto Path

976                   ->  135   TCP

4     System          ->  139   TCP

4     System          ->  445   TCP

1760  GoogleToolbarNotifier->  1025  TCP   C:\Program
Files\Google\GoogleToolbarNotifier\GoogleToolbarNotifier.
exe
1176                  ->  1029  TCP


0     System          ->  123   UDP

0     System          ->  137   UDP

0     System          ->  138   UDP

976                   ->  445   UDP

4     System          ->  500   UDP

1760  GoogleToolbarNotifier->  1900  UDP   C:\Program
Files\Google\GoogleToolbarNotifier\GoogleToolbarNotifier.
exe
0     System          ->  1900  UDP

1176                  ->  4500  UDP
```

Dump B

```
Pid   Process             Port  Proto Path

976                   ->  135   TCP

4     System          ->  139   TCP

4     System          ->  445   TCP

1760  GoogleToolbarNotifier->  1025  TCP   C:\Program
Files\Google\GoogleToolbarNotifier\GoogleToolbarNotifier.
exe
1176                  ->  1029  TCP

1552  Explorer        ->  16016 TCP   C:\WINDOWS\
Explorer.EXE
1552  Explorer        ->  16661 TCP   C:\WINDOWS\
Explorer.EXE
1552  Explorer        ->  28285 TCP   C:\WINDOWS\
Explorer.EXE
1552  Explorer        ->  38530 TCP   C:\WINDOWS\
Explorer.EXE

1552  Explorer        ->  123   UDP   C:\WINDOWS\
Explorer.EXE
1176                  ->  137   UDP

4     System          ->  138   UDP

976                   ->  445   UDP

4     System          ->  500   UDP

1552  Explorer        ->  1900  UDP   C:\WINDOWS\
Explorer.EXE
1760  GoogleToolbarNotifier->  1900  UDP   C:\Program
Files\Google\GoogleToolbarNotifier\GoogleToolbarNotifier.
exe
1552  Explorer        ->  4500  UDP   C:\WINDOWS\
Explorer.EXE
```

Figure 2: Baseline dumps from a clean system and from a possibly infected system.

(http://www.oneysoft.com/fcompare.htm), which highlights changes between dump files of this nature using the File: Compare option. While not perfect, this can be useful for rapid triage of larger dump files compared against baseline dumps.

In Figure 3, *FCompare* highlights the changes between two *F-Port* dumps, showing potentially malicious processes in yellow on the right.

4. TCPView (http://technet.microsoft.com/en-us/sysinternals/bb897437.aspx)

TCPView is a great tool for a quick visual overview of any running processes that are responsible for TCP communications. Since many trojans and other malicious codes attempt to communicate with remote C&Cs via TCP, this is a great way to identify potentially hostile activities. In the example shown in Figure 4, notice that several ports related to the 'non-existent' process as identified by *TCPView*, are all in the listening state.

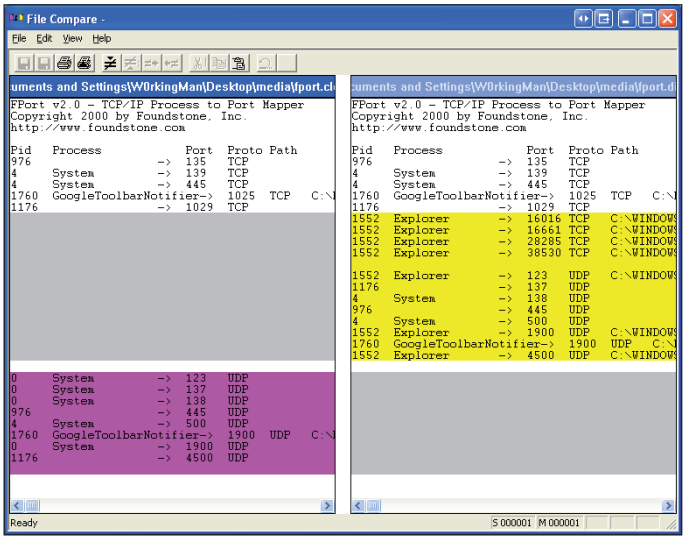


Figure 3: Potentially malicious processes are highlighted in yellow.

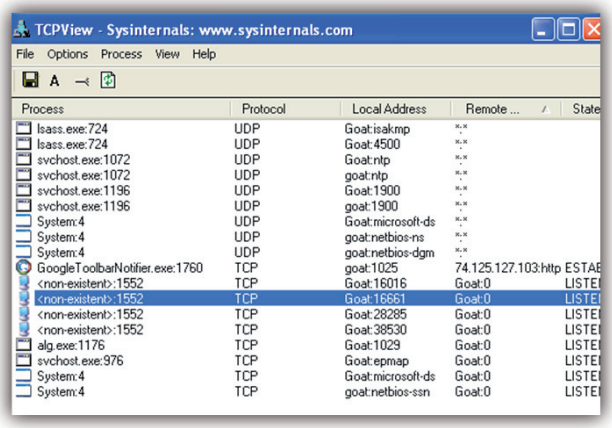


Figure 4: TCPView shows that several ports related to the 'non-existent' process are in the listening state.

TCPView reveals several new 'non-existent' TCP processes related to the Haxdoor rootkit.

5. IceSword (http://www.brothersoft.com/icesword-63677.html)

IceSword highlights any data it believes to be associated with rootkit activity. This can be a great visual when it works properly, as in the example shown in Figure 5 where Haxdoor has injected explorer.exe.

In the example of Haxdoor, *IceSword* is able to show rootkit components in many areas, including but not limited to a hostile process highlighted in red, new processes listening (not in red), system service descriptor table (SSDT) hooks, log/thread creation activity, and hidden files in the *Windows* System32 directory. Analysis of other areas, such as the *Windows* registry, are also possible using this powerful tool.

Comparing a *Windows* file listing of the System32 directory against an *IceSword* file listing of the same directory can also reveal hidden files, as shown in Figure 6. Common locations for concealing malcode are in the System32, *Windows*, and various related directories.

Hint: A good way to compare directories is to sort by created date. Simply right-click on the headers for file listings and select 'Date Created' and then click on the tab after creation to sort in the order desired.

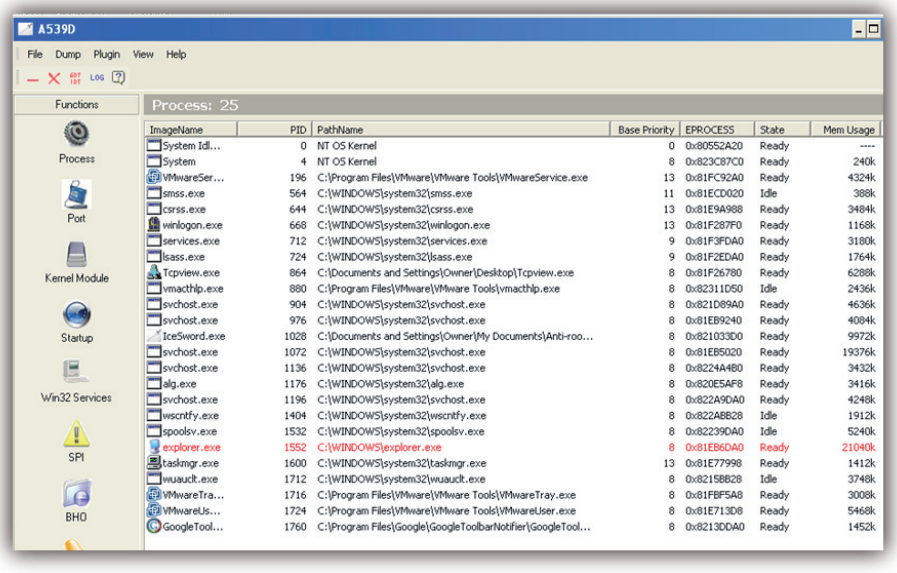


Figure 5: IceSword highlights a malicious rootkit process injected into explorer.exe.

CAPTURE

With triage completed, there may be clues that a rootkit is running on the system. In some cases an incident handler will already have captured hostile image files using tools like *IceSword* and/or forensic techniques. If further investigation is required, capturing physical memory to a file is the next step. Image files are then analysed using the *Volatility Framework* and/or advanced reverse engineering techniques.

Several utilities exist to image physical memory (dump volatile data to a file). Raw image files of physical memory are DD-style copies of the memory but do not contain processor state. This enables an incident handler to compare what is found inside a dump file to what was found at the triage stage. Additionally, processes in memory – including hidden hostile executables – can be extracted from an image file for analysis.

Several tools exist to quickly dump physical memory to an image file. Dumps typically take several minutes and can be quite large: on average 3GB to 4GB and upwards. It is common to dump to the C: drive in order to locate and extract images and other data quickly.

1. Win32dd (http://windd.msuiche.net/)

Win32dd is a free tool that can be used to dump physical memory to a file. It supports *Windows 2000* to *Windows 7* and is capable of producing a full snapshot similar to a *Microsoft* crash dump file.

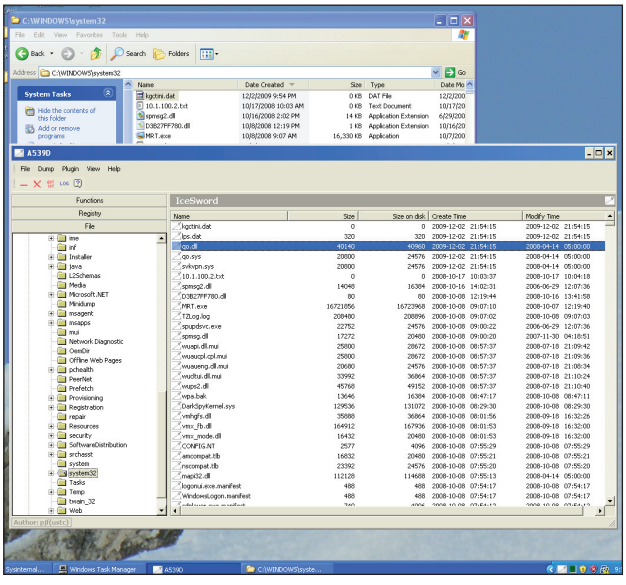


Figure 6: Files appear within IceSword that are not visible within Windows itself, revealing a rootkit running on the system.

Win32dd is a very intuitive, easy-to-use tool that does a great job of imaging quickly. When running it in a CMD window it is wise to navigate to the directory containing the executable to avoid the errors that might appear if the SYS file can't be located (e.g. 'cannot start the driver').

```
win32dd -r C:\filename.dmp
-r = raw memory dump/snapshot
```

2. MDD (http://sourceforge.net/projects/mdd/)

MDD is another tool that can be used to dump physical memory to a file. It is open source, hosted by *SourceForge*. It supports *Windows 2000*, *XP*, *Vista* and *Windows Server*. This program was not very stable in the limited tests I performed, but it did the job well when crashes did not occur.

```
mdd -v -o C:\filename.dmp
-v = verbose; -o = output file followed by path/name
```

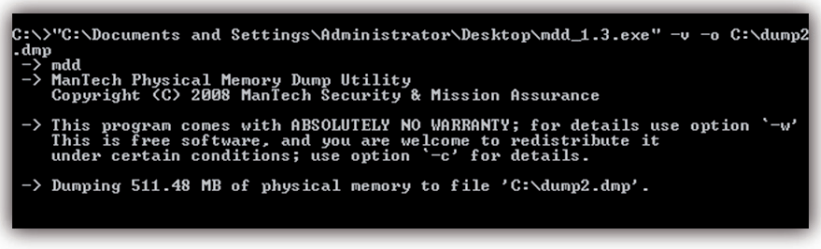


Figure 7: Dumping physical memory to an image file using MDD.

3. Memoryze (http://www.mandiant.com/software/memoryze.htm)

Memoryze is a powerful tool, but it requires a set-up installer to be run before it can be used – this is not usually possible in an incident management situation. Furthermore, the output is best analysed using other *Mandiant* tools that have similar set-up requirements and customized interpretations. This tool is more useful for in-depth investigations or in environments where such tools are used across the enterprise rather than individually for incident handling.

If *Memoryze* is to be installed and used, a directory must be created for dump data. Once such a directory has been created, the tool is able to dump memory using a batch file as shown below:

```
MemoryDD.bat –output <directory_name>
Located at C:\Program Files\Mandiant\Memoryze\
MemoryDD.bat\ in most situations.
```

Mandiant has another tool called *Audit Viewer* which is useful for analysing *Memoryze* images and XML data. A specific Python solution is required to fully install and use the software.

4. VMEM (VMware)

VMware is commonly used to test malicious codes and to research emergent threats. If code runs inside *VMware,* or a virtualized desktop is in use, analysis of a .vmem file is another option for determining what is in memory on a suspect host. Simply click on the *VMware* 'suspend' button to create a '.vmem' file on the host system. Use this image file for advanced memory analysis.

Figure 8: Suspending VMware operating systems creates a .vmem file on the host system.

5. Dcfldd (http://dcfldd.sourceforge.net/)

Dcfldd is yet another method of creating an image file. The syntax can be a bit tricky with forward and backslash conventions:

```
dcfldd if=/dev/mem of=c:\filename.dmp
conv=sync,noerror
```

Volatility Framework Installation

1. Applications:Terminal
2. su -
   Superuser mode
3. CD /home/*CurrentUser*/desktop
   Navigate to the Desktop
4. wget https://www.volatilesystems.com/volatility/1.3/Volatility-1.3_Beta.tar.gz
   Download a copy of the VF.
5. Right-click on the .gz file and select 'Extract Here'
   Extracts the archive.

ANALYSIS

Once physical memory has been captured to an image file you have a huge file with a bunch of data within it waiting to be discovered. Incident handlers should now have a very good idea of what might be malicious, or where to look on a system. The goal now is to use the *Volatility Framework* to extract executables of interest in memory and to compare the image file against earlier triage data in a diff analysis.

Volatility Framework (https://www.volatilesystems.com/default/volatility#overview)

I prefer to use a *Ubuntu* build to install the *Volatility Framework*. Install details are shown in the callout box above. Common installation pitfalls include not entering into SU mode (which is required for installation); downloading to the root directory and wondering where the download is when done (see CD in step 3 to mitigate); and not using Terminal properly to run the Python Volatility file.

Once installation is complete, *Volatility Framework* commands can be run against the image files copied into the analysis system. Open the terminal and navigate to the *Volatility Framework* directory on the desktop. Then enter the following command to view options for the tool:

```
python volatility --help
```

A sample dump of the options provided by the *Volatility Framework* are below. Items in bold are of greatest interest to an incident handler attempting to compare triage data against what is found in the physical memory image file and/or capture of hostile hidden binaries on the system.

usage: volatility cmd [cmd_opts]

Run command cmd with options cmd_opts

For help on a specific command, run 'volatility cmd --help'

Supported internal commands:

`connections`	**Print list of open connections**
`connscan`	Scan for connection objects
`connscan2`	Scan for connection objects (New)
`datetime`	Get date/time information for image
`dlllist`	Print list of loaded dlls for each process
`dmp2raw`	Convert a crash dump to a raw dump
`dmpchk`	Dump crash dump information
`files`	**Print list of open files for each process**
`hibinfo`	Convert hibernation file to linear raw image
`ident`	Identify image properties
`memdmp`	Dump the addressable memory for a process
`memmap`	Print the memory map
`modscan`	Scan for modules
`modscan2`	Scan for module objects (New)
`modules`	Print list of loaded modules
`procdump`	**Dump a process to an executable sample**
`pslist`	**Print list of running processes**
`psscan`	Scan for EPROCESS objects
`psscan2`	Scan for process objects (New)
`raw2dmp`	Convert a raw dump to a crash dump
`regobjkeys`	Print list of open regkeys for each process
`sockets`	Print list of open sockets
`sockscan`	Scan for socket objects
`sockscan2`	Scan for socket objects (New)
`strings`	**Match physical offsets to virtual addresses (may take a while, VERY verbose)**
`thrdscan`	Scan for ETHREAD objects
`thrdscan2`	Scan for thread objects (New)
`vaddump`	Dump the vad sections to files
`vadinfo`	Dump the vad info
`vadwalk`	Walk the vad tree

Supported plug-in commands:

`memmap_ex_2`	Print the memory map
`pslist_ex_1`	Print list running processes
`pslist_ex_3`	Print list running processes
`usrdmp_ex_2`	Dump the address space for a process

Example: `volatility pslist -f /path/to/my/file`

To get specific parameters and help on commands of interest, use the syntax 'python volatility command --help'. For example, 'python volatility pslist --help' produces the following list of options for the pslist command:

```
Usage: pslist [options] (see --help)

Options:

-h, --help show this help message and exit

-f FILENAME, --file=FILENAME (required) XP
SP2 Image file

-b BASE, --base=BASE (optional, otherwise
best guess is made) physical offset (in
hex) of directory table base

-t TYPE, --type=TYPE (optional,
default='auto') identify the image type
(pae, nopae, auto)
```

When running the command 'python volatility pslist –f dump.dmp', where dump.dmp is the file containing physical memory, the *Volatility Framework* provides the sample output shown in Figure 9.

The list in Figure 9 clearly shows Win32dd, FPort and other processes in memory along with their process IDs. Comparing this against triage data and correlating PIDs is the first step in performing advanced memory analysis diff comparisons to discover hidden processes. Once a hostile process is identified, the *Volatility Framework* can be used to dump it to an executable file for further analysis. In the example statement below, a hostile process with the PID value of 1004 is dumped to an executable file for further analysis:

```
python volatility procdump -f dump.dmp -p
1004
```

NOTE: If you don't specify the PID value, *all* processes will be dumped to executable files.

Name	Pid	PPid	Thds	Hnds	Time
System	4	0	55	280	Thu Jan 01 00:00:00 1970
smss.exe	420	4	3	19	Wed Mar 25 17:32:22 2009
csrss.exe	692	420	13	348	Wed Mar 25 17:32:30 2009
winlogon.exe	716	420	19	516	Wed Mar 25 17:32:31 2009
services.exe	760	716	15	248	Wed Mar 25 17:32:32 2009
lsass.exe	772	716	22	342	Wed Mar 25 17:32:33 2009
vmacthlp.exe	936	760	1	25	Wed Mar 25 17:32:33 2009
svchost.exe	952	760	16	195	Wed Mar 25 17:32:34 2009
svchost.exe	1016	760	11	250	Wed Mar 25 17:32:34 2009
svchost.exe	1100	760	57	1292	Wed Mar 25 17:32:34 2009
svchost.exe	1176	760	5	76	Wed Mar 25 17:32:34 2009
svchost.exe	1316	760	13	193	Wed Mar 25 17:32:35 2009
spoolsv.exe	1460	760	11	116	Wed Mar 25 17:32:36 2009
explorer.exe	1680	1632	27	685	Wed Mar 25 17:32:37 2009
VMwareTray.exe	212	1680	1	29	Wed Mar 25 17:32:40 2009
VMwareUser.exe	224	1680	8	115	Wed Mar 25 17:32:40 2009
msmsgs.exe	244	1680	2	155	Wed Mar 25 17:32:41 2009
VMwareService.e	668	760	3	147	Wed Mar 25 17:32:43 2009
alg.exe	2008	760	5	103	Wed Mar 25 17:32:50 2009
wscntfy.exe	308	1100	1	28	Wed Mar 25 17:32:50 2009
procexp.exe	1188	1680	0	-1	Fri Oct 09 20:18:43 2009
aports.exe	660	1680	0	-1	Fri Oct 09 20:18:50 2009
verclsid.exe	240	1680	0	-1	Fri Oct 09 20:18:56 2009
verclsid.exe	1692	1680	0	-1	Fri Oct 09 20:18:56 2009
cmd.exe	740	1680	0	-1	Fri Oct 09 20:18:57 2009
Fport.exe	1072	740	0	-1	Fri Oct 09 20:19:31 2009
verclsid.exe	976	1680	0	-1	Fri Oct 09 20:20:04 2009
verclsid.exe	1860	1680	0	-1	Fri Oct 09 20:20:04 2009
cmd.exe	1640	1680	0	-1	Fri Oct 09 20:29:35 2009
cmd.exe	2004	1680	2	47	Fri Oct 09 20:29:41 2009
win32dd.exe	1480	2004	2	44	Fri Oct 09 20:29:56 2009

Figure 9: Sample output provided by the Volatility Framework.

Extracted binaries enable researchers to scan files to see if they are infected by malicious code, perform reverse engineering, and more. Investigations should start with a strings analysis. Most processes are not obfuscated or packed while in memory, thus allowing plenty of opportunities for strings analysis. In some cases URLs and other data can easily be seen when looking at strings of executables captured from a dump file.

Additional work can now be done both on the original infected system and in the analysis of memory data and extracted executables. Going back to the original system, anti-rootkit and forensic techniques can now be used to identify and extract specific image files of interest. Additionally, advanced analysis of extracted raw image data frequently leads to discoveries that can impact live system analysis and reverse engineering, URLs, remote server investigations and related abuse data, and more.

In next month's issue, we will take a detailed look at the use of advanced memory analysis on a system infected with Haxdoor.

COMPARATIVE REVIEW

NOVELL SUSE LINUX ENTERPRISE SERVER 11

John Hawes

Our annual excursion to the calm and balmy shores of *Linux* comes at the perfect time for us. The stresses and strains of the previous comparative – featuring a record number of participants on the shiny new *Windows 7* platform – are gradually fading to a glorious, if rather painful memory, while the prospect looms of the *XP* comparative in the spring, which promises an even larger field of products to slog through. Sandwiched between these two behemoths, the *Linux* test provides a welcome moment of respite.

With far fewer products available for the *Linux* platform than for most others we test on, we always expect a quiet month, but it appeared from the start that our choice of distribution would make for an even smaller test than anticipated. Although *Novell*'s *SUSE Linux* is one of the most well funded and heavily marketed commercial server distributions, and its most recent edition (version 11) was released some nine months prior to the test deadline, it still presented enough of a challenge to put off entries from several of those normally expected to take part in our tests. The two largest security firms, *Symantec* and *McAfee*, were unable to provide products supporting the platform, although *Symantec* promises a new edition with full support due for release very soon.

Several others also decided not to take part due to timing issues and new releases pending.

Another major vendor also opted to skip this test, again with a significant upgrade to its *Linux* product close on the horizon but not quite ready for testing. This decision marks something of the end of an era: *VB* has been running VB100 comparative reviews since January 1998, more or less every two months, with a total of 67 sets of results published so far – this month's test will be the first time that *Kaspersky Lab* has not submitted a product. A sad day indeed.

However, some nine products did make their way to us in time for the test deadline, with most of the remaining regular participants present and no further surprises. Small numbers do not necessarily make for a simple test of course, and previous experience of *Linux* tests has led us to expect all manner of opaque and confusing installation procedures, unusual and esoteric implementation and incomplete, inadequate and even well-hidden documentation. Hoping for a smooth and simple test (but as always prepared for the worst), we settled ourselves into the test lab for what was guaranteed to be an interesting month.

PLATFORM AND TEST SETS

Novell's acquisition of the *SUSE* distribution, announced in 2003 and finalized in early 2004, brought *SUSE* to the highest level of commercially supported *Linux* flavours. Always among the market leaders in Europe, it now stands as one of few likely challengers for *Red Hat* in the business sector. Successive releases have brought ever greater stability, completeness of vision and ease of use. The slick installer and the advanced Yast configuration tool bring most tasks involved in setting up and running a system, desktop or server within the grasp of even the most modest administrator. The platform has long been a favourite in the *VB* lab, with the fully open source variant *OpenSUSE* running on many of the servers that support our test networks. The more sober server edition is also the platform of choice for our anti-spam test network, and is selected by default if solution providers have no preference. So the task of preparing the test systems was a straightforward one. Installing and cloning systems was a fast and easy process, and few adjustments were required beyond pointing a few network shares to the right places and sharing the storage areas for the test sets ready for on-access testing. A client system, running *Windows XP SP3*, was set up with these shares mounted as network drives, with the standard set of scripts to run the on-access tests, and things were ready to go.

Putting the test sets in place proved similarly free from problems. The latest additions to the WildList were reasonably unremarkable, with yet more Koobface and OnlineGames variants continuing to dominate the list, and the old guard – the reams of Netsky and Mytob variants which once held sway over the list – continuing to decline. The most significant items on the list remain the complex W32/Virut strains, of which yet another new variant was added in recent months; as usual, our automated replicator churned out several thousand examples ready to challenge the products' detection abilities to the utmost.

Elsewhere in the test sets everything was much as normal. The trojan set was slightly larger than usual thanks to a longer than average gap since the last test and some further expansion of our sample-gathering efforts. Meanwhile, the RAP sets were kept to a reasonable size thanks to a notable decrease in the number of incoming samples over the Christmas and New Year period, when most of the samples were taken in (probably due to various labs taking time off over the festive season); numbers quickly climbed back to previous levels a few weeks into 2010, with many sources providing extra large bundles to catch up with backlogs.

The speed sets were left largely unchanged, other than a little cleaning of inappropriate files which had slipped in, and the clean set had a fairly average number of additions,

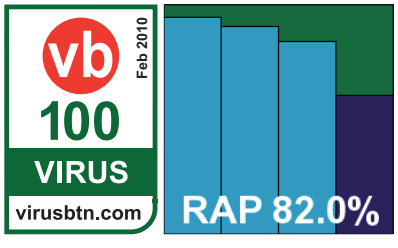
On-demand tests	WildList viruses		Worms & bots		Polymorphic viruses		Trojans		Clean sets	
	Missed	%	Missed	%	Missed	%	Missed	%	FP	Susp.
Alwil avast!	0	100.00%	0	100.00%	8	99.39%	1640	92.60%	0	0
Avira AntiVir	0	100.00%	0	100.00%	0	100.00%	373	98.32%	0	0
CA Threat Manager	0	100.00%	0	100.00%	959	92.00%	11632	47.52%	0	0
eScan	0	100.00%	0	100.00%	0	100.00%	1719	92.24%	0	0
ESET Security	0	100.00%	0	100.00%	0	100.00%	707	96.81%	0	9
Frisk F-PROT	0	100.00%	0	100.00%	0	100.00%	4442	79.96%	0	0
Quick Heal	1	99.99996%	0	100.00%	0	100.00%	2715	87.75%	0	0
Sophos Anti-Virus	0	100.00%	0	100.00%	0	100.00%	1604	92.76%	0	3
VirusBuster	0	100.00%	0	100.00%	193	89.10%	2461	88.90%	2	0

most of which were popular freeware utilities and some software provided free with magazines and hardware acquired by the lab in recent months. A dedicated set of *Linux* files was also compiled for the speed tests, taken from the /bin, /sbin, /etc, /opt and /usr folders of one of the test machines. As usual for the speed tests, scanning speeds using both default settings and 'full' settings were recorded – where necessary, settings were increased to include all file types with no size limit, scanning archives internally to the highest available depth. A product's times were counted in the full column if files with non-standard extensions were detected and, for the archive set, if at least four of the eight archive types checked were scanned to a depth of at least four levels. Thanks to the fair number of compressed files in the *Linux* speed set, this was treated in the same way as the archive set for this month's test.

With everything set up and more or less in order, we got down to business.

Alwil avast! for Linux 3.2.0_rc

ItW	100.00%	**Polymorphic**	99.39%
ItW (o/a)	100.00%	**Trojans**	92.60%
Worms & bots	100.00%	**False positives**	0

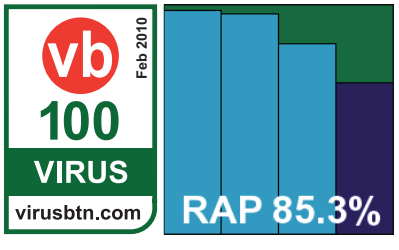


Alwil's product was provided as a selection of RPM installer files, along with some instructions on compiling and installing the *Dazuko* filtering module required by the on-access scanner. Several steps were required, but thanks to the clear instructions everything ran through smoothly and we were soon underway with our first run through the tests.

Running of scans from the command line and operation of the on-access guard proved logical and simple, following pretty simple and predictable formats, and with decent speeds across all sets, results were gathered in pleasingly short order. Detection rates were as excellent as ever, with scanning speeds even more impressive in both modes. No problems were encountered in the WildList or any of the clean sets, and *Alwil* is duly awarded another VB100 for its collection.

Avira AntiVir Linux Server Professional 3.0.5

ItW	100.00%	**Polymorphic**	100.00%
ItW (o/a)	100.00%	**Trojans**	98.32%
Worms & bots	100.00%	**False positives**	0



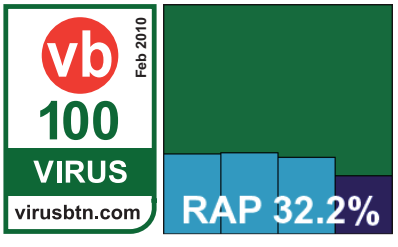
Avira's *Linux* solution uses the *Dazuko* filter module in a slightly different fashion, but provides it pre-compiled and ready to go with just a minor tweak required to one of the set-up scripts. Installation is well automated and lucid, and documentation is similarly clear. Operation and control of the product is thus straightforward, and with good speeds throughout, testing once again took little time or effort.

Some truly superb detection figures were achieved, nearing perfection in the trojans set and pretty impressive in the RAP sets. With the WildList and clean sets causing no difficulties, *Avira* also comfortably earns another VB100.

On-access tests	WildList viruses		Worms & bots		Polymorphic viruses		Trojans	
	Missed	%	Missed	%	Missed	%	Missed	%
Alwil avast!	0	100.00%	0	100.00%	8	99.39%	1453	93.44%
Avira AntiVir	0	100.00%	0	100.00%	0	100.00%	373	98.32%
CA Threat Manager	0	100.00%	0	100.00%	959	92.00%	11632	47.52%
eScan	5	99.48%	4	99.81%	0	100.00%	1360	93.86%
ESET Security	0	100.00%	0	100.00%	0	100.00%	751	96.61%
Frisk F-PROT	0	100.00%	0	100.00%	0	100.00%	4448	79.93%
Quick Heal	1	99.99996%	0	100.00%	0	100.00%	9204	58.48%
Sophos Anti-Virus	0	100.00%	3	99.86%	0	100.00%	1612	92.73%
VirusBuster	0	100.00%	0	100.00%	193	89.10%	2444	88.97%

CA Threat Manager 8.1.5379.0

ItW	100.00%	**Polymorphic**	92.00%
ItW (o/a)	100.00%	**Trojans**	47.52%
Worms & bots	100.00%	**False positives**	0

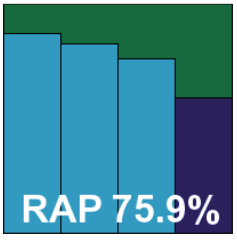


CA's product has a rather more involved and complex set-up procedure, but ample instructions were provided and the product was up and running without much ado. A GUI is provided, accessible from a built-in web server, which closely resembles that seen in numerous *Windows* tests over the last few years. This proved to offer all the functions required, but not in great depth and without the finer control provided by command-line and configuration file set-ups – the preferred method of more serious *Linux* admins, particularly at the server level. A command-line tool for on-demand scanning is provided, but seemed unwilling to work for us without deeper research, so we mostly made do with the GUI. This proved a rather frustrating path, as the interface was flaky in the extreme, with around one click in five (and sometimes as many as one in three) producing a page error and requiring a refresh of the browser and a retry at configuring. Nevertheless, we got there in the end, with the only lasting problem being an apparent lack of archive scanning on access despite options to enable it in the interface – a problem we have noted many times previously with the *Windows* version of the same product.

Speeds were as excellent as we have come to expect from *CA*'s remarkably quick engine, and while detection rates in the trojans and RAP sets were rather disappointing, no problems were encountered in the WildList set and no false positives turned up in the clean sets, thus qualifying *CA* for a VB100 award.

eScan for Linux File Servers 3.3.-3.sles11

ItW	100.00%	**Polymorphic**	100.00%
ItW (o/a)	99.48%	**Trojans**	92.24%
Worms & bots	100.00%	**False positives**	0



The *Linux* version of *eScan* is another complete and professional server package, again with a web-accessible GUI as well as a desktop interface for running on-demand scans, but where possible we went with the command line to enable better automation of tasks. Installation was fairly painless, with a set of RPM installers to run and the on-access protection provided on *Samba* shares with some small additions to make to the *Samba* configuration file. All of this is clearly documented in an accompanying PDF manual.

Scanning speeds were not the fastest, partly thanks to some very thorough default settings, but detection rates seemed generally good. However, on checking the results we found a number of unexpected misses in the WildList and worms and bots sets, with files not spotted on access but easily noted on demand. Deeper analysis showed that the on-access scanner is set to ignore files larger than 2048KB by default – presumably to mitigate slowdowns caused by the thorough scanning. Retrying the speed tests with the limit disabled resulted in numerous problems, including the scanning engine daemon shutting down and disabling all access to the *Samba* share. Since several samples in

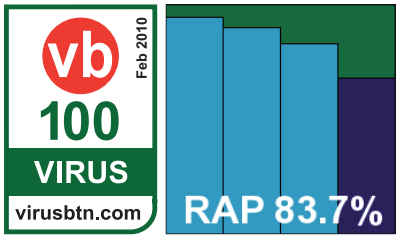
On-demand throughput (Time = s; Throughput = MB/s)	Archive files				Binaries and system files				Linux files				Media and documents				Other file types			
	Default settings		All files		Default settings		All files		Default settings		All files		Default settings		All files		Default settings		All files	
	Time	Thr. put	Time	Thr. put	Time	Thr. put	Time	Thr. put	Time	Thr. put	Time	Thr. put	Time	Thr. put	Time	Thr. put	Time	Thr. put	Time	Thr. put
Alwil avast!	251	11.59	479	6.07	248	19.84	263	18.74	422	8.31	443	7.92	265	13.26	260	13.50	210	5.14	221	4.91
Avira AntiVir	769	3.78	808	3.60	264	18.66	254	19.40	1340	2.62	1438	2.44	313	11.20	328	10.68	267	4.05	242	4.47
CA Threat Manager	1200	2.42	1200	2.42	273	18.04	273	18.04	514	6.83	514	6.83	288	12.18	288	12.18	223	4.85	223	4.85
eScan	1135	2.56	1135	2.56	326	15.12	326	15.12	1198	2.93	1198	2.93	452	7.77	452	7.77	317	3.41	317	3.41
ESET Security	658	4.42	658	4.42	488	10.09	488	10.09	1533	2.29	1533	2.29	319	11.02	319	11.02	233	4.64	233	4.64
Frisk F-PROT	399	7.29	399	7.29	483	10.19	483	10.19	325	10.79	325	10.79	259	13.54	259	13.54	167	6.49	167	6.49
Quick Heal	1517	1.92	1517	1.92	294	16.76	294	16.76	1246	2.82	1246	2.82	568	6.17	568	6.17	320	3.38	320	3.38
Sophos Anti-Virus	32	91.54	2606	1.12	418	11.79	644	7.65	321	10.92	1139	3.08	330	10.63	316	11.11	255	4.24	244	4.43
VirusBuster	370	7.86	1238	2.35	674	7.31	674	7.31	1226	2.86	1689	2.08	506	6.93	506	6.93	254	4.26	254	4.26

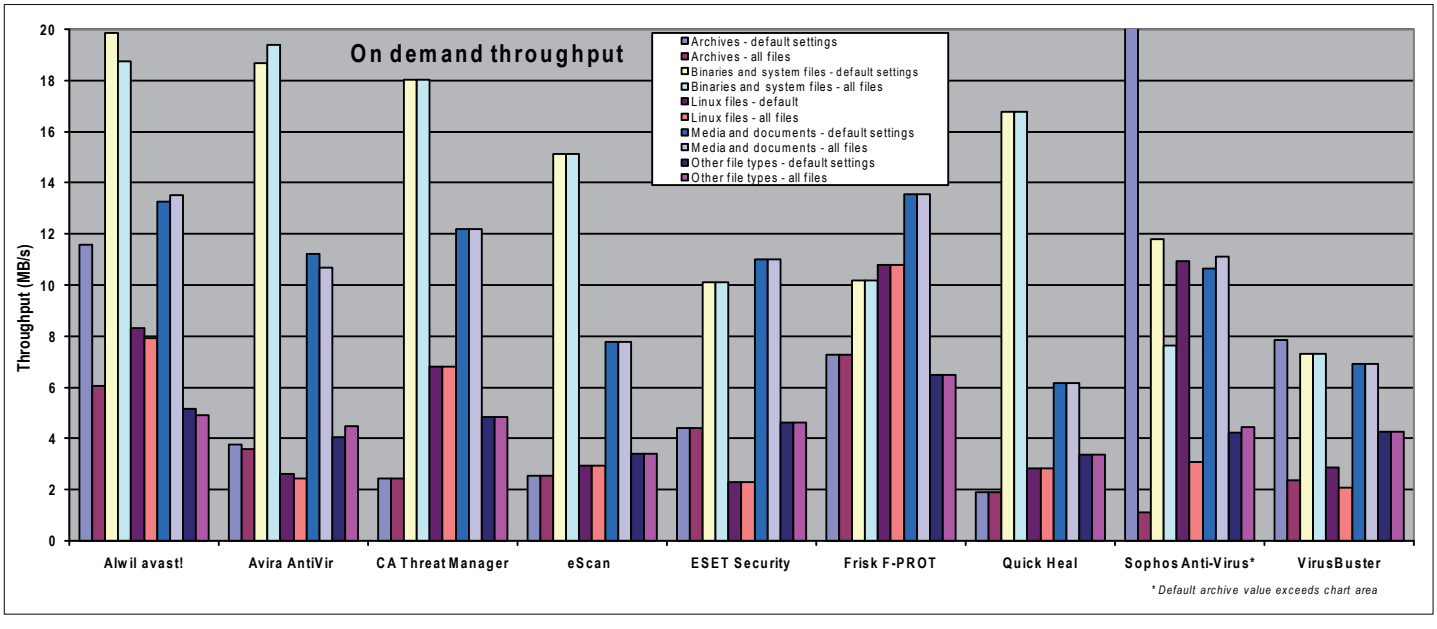
the WildList – including a nasty W32/Bagle worm – were larger than 2MB and thus ignored in the default setting, *eScan* is denied a VB100 award this month.

ESET Security for Linux 3.0.15

ItW	100.00%	**Polymorphic**	100.00%
ItW (o/a)	100.00%	**Trojans**	96.81%
Worms & bots	100.00%	**False positives**	0

ESET also offers some nice simple install scripts – not as straightforward as some, but still fairly easy to operate. On-access scanning can be provided either using the *Dazuko* module, allowing full system protection,





Archive scanning		ACE	CAB	EXE-ZIP	JAR	LZH	RAR	TGZ	ZIP	EXT*
Alwil avast!	OD	X/√	X/√	√	√	X/√	X/√	√	√	√
	OA	√	√	√	√	√	√	√	√	√
Avira AntiVir	OD	2	X/√	√	X/√	X/√	X/√	√	√	√
	OA	2	√	√	√	√	√	√	√	√
CA Threat Manager	OD	X	√	X	√	√	√	√	√	√
	OA	X	X	X	1	X	X	X	1	√
eScan	OD	√	√	8	√	√	√	8	√	√
	OA	√	√	8	√	√	√	8	√	√
ESET Security	OD	√	√	√	√	√	√	5/√	√	√
	OA	√	√	√	√	√	√	5	√	√
Frisk F-PROT	OD	5	5	5	5	√	5	2	5	√
	OA	√	√	√	√	√	√	√	√	√
Quick Heal	OD	X	√	X	√	X	√	X	√	√
	OA	2	X	X	X	X	X	X	X	√
Sophos Anti-Virus	OD	X	X/5	X/5	X/5	X/5	X/5	X/5	X/5	X/√
	OA	X	X/√	X/7	X/√	X/√	X/√	X/7	X/√	√
VirusBuster	OD	2	√	√	X/√	X	√	√	√	X/√
	OA	X	X	X	X	X	X	X	X	√

Key: X - Archive not scanned; X/√ - Default settings/thorough settings; √ - Archives scanned to depth of 10 or more levels; [1-9] - Archives scanned to limited depth; EXT* - Eicar test file with random extension; All others - detection of Eicar test file embedded in archive nested up to 10 levels

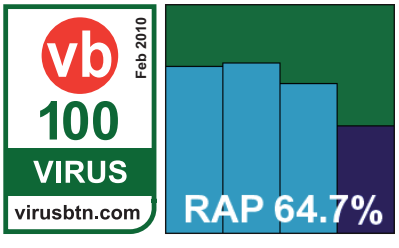
or on *Samba* shares only; for simplicity we opted to use this method, and again it proved simple to set up and configure.

On-demand scanning speeds were pretty reasonable, and on-access overheads not too heavy, despite some pretty intensive default scanning levels. Detection rates were quite excellent, with the only problem encountered in the RAP sets, where a couple of files caused the engine to trip up with a segmentation fault error message. With these moved out of the way, RAP scores proved just as impressive as those in the main sets, and the clean sets threw up only a few (fairly accurate) warnings of potentially unwanted adware-type products (mostly toolbars included 'free' with some of the trialware products added this month). With no full blown false positives, and the WildList handled with ease, *ESET* earns another VB100 award to add to its impressive haul.

Frisk F-PROT AntiVirus for Linux FileServers 6.3.3.5015

ItW	100.00%	**Polymorphic**	100.00%
ItW (o/a)	100.00%	**Trojans**	79.96%
Worms & bots	100.00%	**False positives**	0

Frisk's *Linux* product, like its *Windows* versions, is simple in the extreme, with most of it quite happy to run from wherever the initial zip is unpacked, but a Perl installer script is provided to simplify the set-up process.



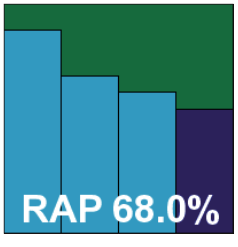
Again, the choice of *Dazuko* or *Samba*-based on-access protection is offered, and again we opted for the *Samba* method as all on-access tests were being run from a *Windows* client. Everything was up and running fairly simply despite a lack of clear documentation.

On-demand scanning speeds were excellent, and on-access overheads were not bad either. Detection rates proved decent, if not overwhelming. The clean sets and the WildList presented no difficulties, and as a result *Frisk* also earns itself another VB100 award.

File access lag time (Time = s; Lag = s/MB)	Archive files				Binaries and system files				Linux files				Media and documents				Other file types			
	Default settings		All files		Default settings		All files		Default settings		All files		Default settings		All files		Default settings		All files	
	Time	Lag	Time	Lag	Time	Lag	Time	Lag	Time	Lag	Time	Lag	Time	Lag	Time	Lag	Time	Lag	Time	Lag
Alwil avast!	1366	0.26	1366	0.26	1406	0.06	1406	0.06	3806	0.17	3806	0.17	919	0.06	919	0.06	506	0.08	506	0.08
Avira AntiVir	2094	0.51	2094	0.51	1275	0.03	1275	0.03	4366	0.33	4366	0.33	917	0.06	917	0.06	512	0.09	512	0.09
CA Threat Manager	759	0.05	NA	NA	1439	0.06	1439	0.06	5646	0.69	NA	NA	1493	0.30	1493	0.30	763	0.32	763	0.32
eScan	683	0.03	1882	0.44	1399	0.06	1414	0.06	9456	1.78	9865	1.89	1846	0.45	1954	0.49	990	0.53	997	0.53
ESET Security	1187	0.20	1187	0.20	1961	0.17	1961	0.17	4237	0.29	4237	0.29	1041	0.11	1041	0.11	617	0.18	617	0.18
Frisk F-PROT	977	0.13	977	0.13	1555	0.09	1555	0.09	4950	0.49	4950	0.49	932	0.07	932	0.07	492	0.07	492	0.07
Quick Heal	669	0.02	NA	NA	1420	0.06	1420	0.06	4432	0.34	NA	NA	1163	0.16	1163	0.16	587	0.15	587	0.15
Sophos Anti-Virus	619	0.00	1842	0.42	1434	0.06	1469	0.07	3860	0.18	4044	0.23	943	0.07	940	0.07	584	0.15	584	0.15
VirusBuster	634	0.01	NA	NA	1726	0.12	1726	0.12	4997	0.51	NA	NA	956	0.08	956	0.08	537	0.11	537	0.11

Quick Heal for Linux 11.00

ItW	99.99%	**Polymorphic**	100.00%
ItW (o/a)	99.99%	**Trojans**	87.75%
Worms & bots	100.00%	**False positives**	0



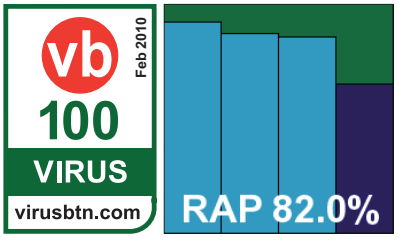
Quick Heal's product was one of few to have dependencies, in the form of a compatibility library for some older C++ code, but this presented little problem. *Dazuko* was the on-access filtering method of choice, with again a slightly different implementation, but it proved no problem to set up and get running. This needs to be done manually before the installer script is run, but with it in place everything else runs like clockwork, with helpful and descriptive comments and instructions provided.

As ever, *Quick Heal* sped through the tests, with some superb times recorded in the on-access tests, the protection barely registering. This effect may have been helped by a lack of archive scanning on access – something which seemed impossible to activate in the rather minimal configuration system. Detection rates were pretty good on demand, although a notable difference between on-demand and on-access scores hinted at either wildly different settings or some stability issues with the on-access implementation. Further investigation and re-tests showed more variations in detection and speeds, with the protection appearing rather unstable on access. The fast speeds recorded may be a side effect of this uncertain application of scanning to files being rapidly accessed.

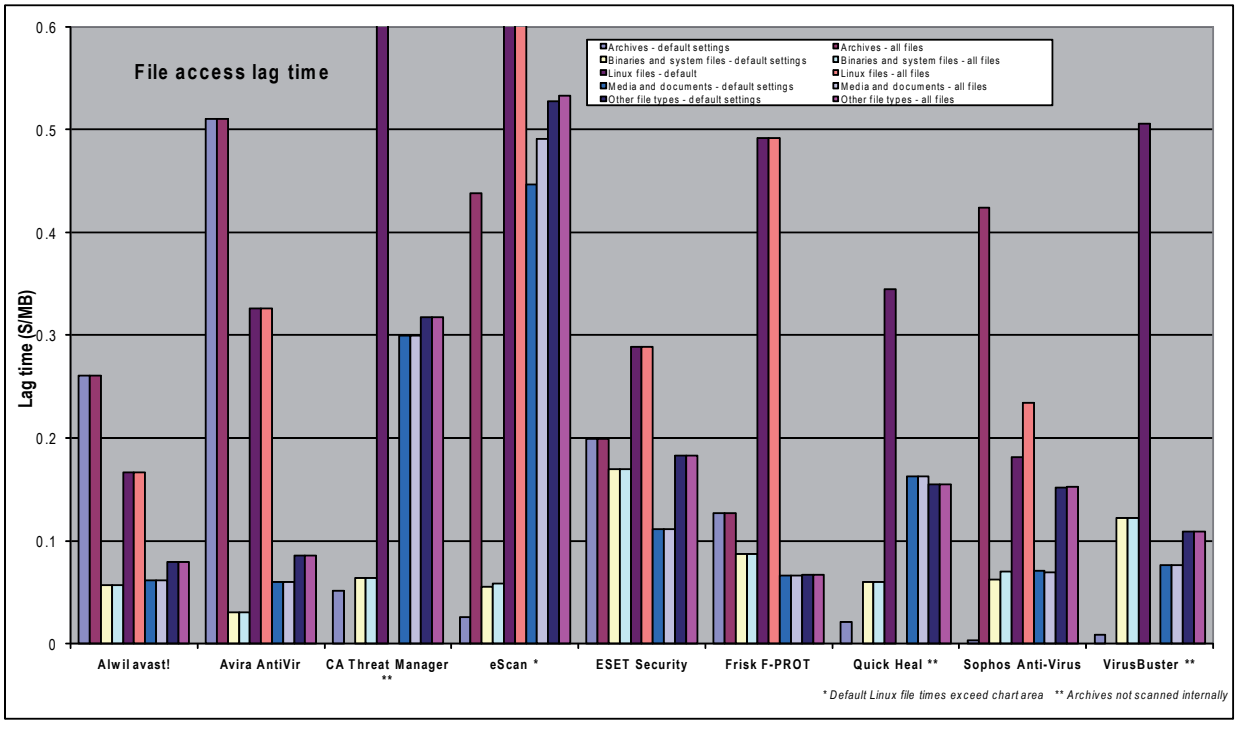
Elsewhere, a fairly steep decline was observed across the RAP sets, although with a solid starting point the overall average was decent. The clean sets were handled accurately, but in the WildList set a single sample of W32/Virut, from the latest strain added to the list, went undetected both on demand and on access, and *Quick Heal* does not quite make the required grade for VB100 certification this month.

Sophos Anti-Virus for Linux 6.7.3

ItW	100.00%	**Polymorphic**	100.00%
ItW (o/a)	100.00%	**Trojans**	92.76%
Worms & bots	100.00%	**False positives**	0



Sophos's product had one of the slickest installation processes, running smoothly and cleanly through the required steps, including the selection and implementation of its own 'Talpa' on-access hooking set-up. With everything set up and operational in double-quick

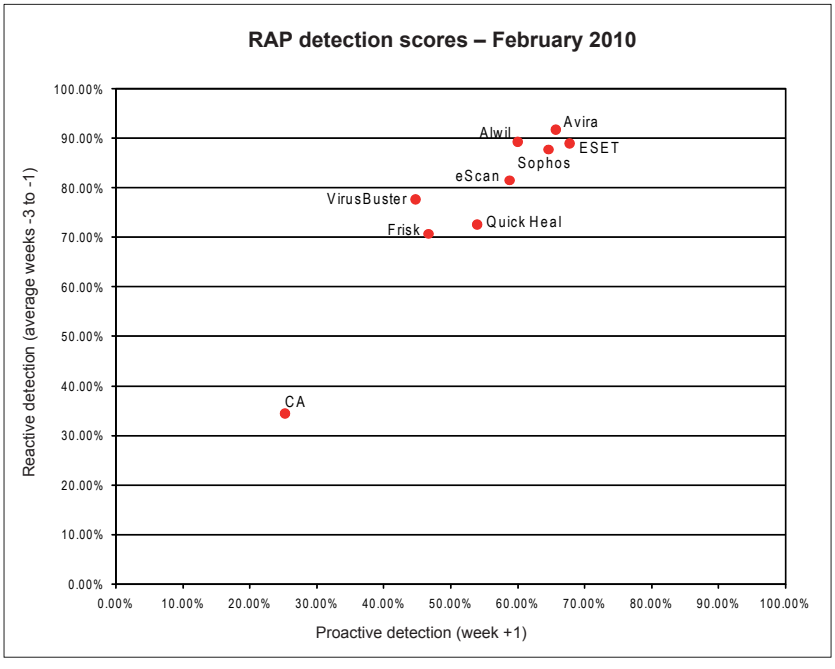


time, testing sped through thanks to pleasantly sensible and standardized command-line options for the on-demand scanner, and a slightly more fiddly but well-documented control system for the on-access component.

Scanning speeds were not bad in either mode, and detection rates proved very solid throughout the sets. However, a few anomalies were observed in the on-demand scan; relying on extension lists to decide whether or not to scan files, it appeared that at least one viable executable extension had been missed off the list. When testing and replicating samples, we endeavour to capture copies of each sample with all the extensions it uses while spreading, to check for just such errors. With several pieces of malware using this extension to conceal spreading files from less sophisticated users, *Sophos* is lucky to have covered this month's WildList without issues. As it is, several samples in the worms and bots set – all of which had been retired from the WildList in recent months – went undetected with the default settings.

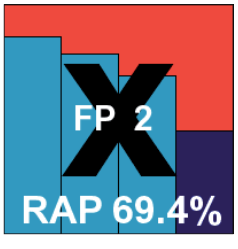
A VB100 award is earned, but we hope to see the error fixed promptly.

Reactive And Proactive (RAP) detection scores	Reactive			Reactive average	Proactive	Overall average
	week -3	week -2	week -1		week +1	
Alwil avast!	94.22%	90.30%	83.54%	89.35%	60.07%	82.03%
Avira AntiVir	97.38%	95.60%	82.51%	91.83%	65.76%	85.31%
CA Threat Manager	34.79%	35.10%	33.48%	34.46%	25.24%	32.15%
eScan	86.89%	82.04%	75.68%	81.54%	58.81%	75.86%
ESET Security	94.25%	89.98%	82.76%	89.00%	67.81%	83.70%
Frisk F-PROT	72.93%	74.11%	65.19%	70.74%	46.68%	64.73%
Quick Heal	88.42%	68.03%	61.43%	72.62%	53.99%	67.97%
Sophos Anti-Virus	91.69%	86.57%	85.04%	87.77%	64.62%	81.98%
VirusBuster	85.69%	78.49%	68.81%	77.66%	44.77%	69.44%



VirusBuster SambaShield for Linux 1.2.1_3-1.2.2_4

ItW	100.00%	**Polymorphic**	89.10%
ItW (o/a)	100.00%	**Trojans**	88.90%
Worms & bots	100.00%	**False positives**	2



As the name hints, *VirusBuster*'s *SambaShield* provides protection for *Samba* shares, which is just what is required for our test. The set-up process is simple and quick, with a nice installer script putting things in place and making the required tweaks to the *Samba* configuration file. Some rummaging around is subsequently required to find the scanner and other components. With these located, some fairly logical controls are provided for the on-access scanner, while the on-demand portion seemed less deeply configurable.

Testing zipped along nicely, and scanning speeds were pretty good in general, although slower than many at handling the large number of small files in the *Linux* speed set. Detection rates showed further improvements to those noted in recent months, with a reasonable decline across the RAP sets. The WildList, with its many tricky Virut samples, was handled with aplomb, but in the clean sets a couple of files – one from *Roxio* and another from an *AOL* set-up CD – were alerted on as trojans, thus spoiling *VirusBuster*'s chances of a VB100 award this month.

CONCLUSIONS

This month we saw some products with tricky and esoteric installation and control systems among a field dominated by a pleasant level of clarity and good design. We saw slow scanning times and heavy on-access lags, although most products were reasonably lightweight and nimble. We saw some disappointing detection rates alongside some highly impressive scores. Having expected a high pass rate – perhaps even a clean sweep – we saw problems with false positives, missed polymorphic samples in the WildList set, and an unfortunate default setting causing products to miss the required standard for the award. All in all, a bit of a mixed bag.

A few interesting trends and patterns were noted. Unlike most of our tests on the *Windows* platform, where on-demand scores are almost invariably higher than those recorded on access, we saw several products doing less well with their on-demand scans. This, of course, is due to the way the products are operated, with command-line scanners often defaulting to less thorough defaults than graphical ones, giving the operator more freedom and control to design scans as required. In the *Linux* context, the concept of 'default' is perhaps a little less exact than in most of our tests, although of course all scanners have their own list of automatically enabled options.

The RAP tests produced some interesting results once again, with most products taking up familiar positions in the RAP quadrant. As our automation systems improve we are slowly increasing the size of the RAP sets, as well as fine-tuning their correlation with prevalence and telemetry data and thus their relevance, and hopefully this will continue to increase the value of the data provided.

We also hope soon to implement a long-planned series of improvements in the polymorphic and worms-and-bots sets, as well as the redesign and rebuilding of our clean, speed and false positive sets. Much of this should be in place in time for the next test – for which we expect a vastly expanded field of products, including a number of newcomers.

Technical details:

All products were tested on identical systems with *AMD Athlon64 X2* Dual Core 5200+ processors, 2GB RAM, dual 80GB and 400GB hard drives. Test systems were running *Novell SUSE Linux Enterprise Server 11*, 32-bit edition, with *Linux Kernel 2.6.27.19*. On access tests were performed from clients running *Microsoft Windows XP, Service Pack 3*, accessing network shares exported using *Samba 3.2.7*.

END NOTES & NEWS

RSA Conference 2010 will be held 1–5 March 2010 in San Francisco, CA, USA. For details see http://www.rsaconference.com/.

The 7th Annual Enterprise Security Conference will take place 3–4 March 2010 in Kuala Lumpur, Malaysia. For details see http://www.acnergy.com/EntSec2010.htm.

Security Summit Milan takes place 16–18 March 2010 in Milan, Italy (in Italian). For details see https://www.securitysummit.it/.

The 11th annual CanSecWest conference will be held 22–26 March 2010 in Vancouver, Canada. For more details see http://cansecwest.com/.

The MIT Spam Conference 2010 is scheduled to take place 25–26 March 2010. For details see http://projects.csail.mit.edu/spamconf/.

Black Hat Europe 2010 takes place 12–15 April 2010 in Barcelona, Spain. For details see http://www.blackhat.com/.

The New York Computer Forensics Show will be held 19–20 April 2010 in New York, NY, USA. For more information see http://www.computerforensicshow.com/.

Infosecurity Europe 2010 will take place 27–29 April 2010 in London, UK. For more details see http://www.infosec.co.uk/.

The 19th EICAR conference will be held 10–11 May 2010 in Paris, France with the theme 'ICT security: quo vadis?'. For more information see http://www.eicar.org/conference/.

The fourth annual Counter-eCrime Operations Summit (CeCOS IV) will take place 11–13 May 2010 in Sãu Paulo, Brazil. For details see http://www.apwg.org/events/2010_opSummit.html.

The International Secure Systems Development Conference (ISSD) takes place 20–21 May 2010 in London, UK. For details see http://issdconference.com/.

NISC11 will be held 19–21 May 2010 in St Andrews, Scotland. Interest in attending can be registered at http://nisc.org.uk/.

CARO 2010, the 4th International CARO workshop will take place 26–27 May 2010 in Helsinki, Finland. The workshop will focus on the topic of 'Big Numbers'. For more information see http://www.caro2010.org/.

Security Summit Rome takes place 9–10 June 2010 in Rome, Italy (in Italian). For details see https://www.securitysummit.it/.

The 22nd Annual FIRST Conference on Computer Security Incident Handling takes place 13–18 June 2010 in Miami, FL, USA. For more details see http://conference.first.org/.

The Seventh International Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA) will take place 8–9 July 2010 in Bonn, Germany. For more information see http://www.dimva.org/dimva2010/.

CEAS 2010 – the 7th annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference – will be held 13–14 July 2010 in Redmond, WA, USA. A call for papers has been issued, with a deadline for submissions of 26 March. As in previous years the conference will run a 'spam challenge'. For details see http://ceas.cc/.

Black Hat USA 2010 takes place 24–29 July 2010 in Las Vegas, NV, USA. DEFCON 18 follows the Black Hat event, taking place 29 July to 1 August, also in Las Vegas. For more information see http://www.blackhat.com/ and http://www.defcon.org/.

The 19th USENIX Security Symposium will take place 11–13 August 2010 in Washington, DC, USA. For more details see http://usenix.org/.

VB2010 will take place 29 September to 1 October 2010 in Vancouver, Canada. *VB* is currently seeking submissions from those wishing to present papers at the conference, see p.10. For details of sponsorship opportunities and any other queries relating to VB2010, please contact conference@virusbtn.com.

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)1865 543153
Email: editorial@virusbtn.com Web: http://www.virusbtn.com/