NOVEMBER 2007

virus BULLETIN

Fighting malware and spam

CONTENTS

ISSN 1749-7027

IN THIS ISSUE

FULL KERNEL MALWARE

Trojan.srizbi is the first example seen in the wild of a complex piece of malware that operates fully from kernel mode. Kimmo Kasslin and Elia Florio provide a detailed analysis.

page 5

STORMING THE INTERNET

Pierre-Marc Bureau and Andrew Lee examine the evolution of W32/Nuwar, the 'Storm worm', from its rather humble beginnings as a minor malware threat into what is probably one of the most bleeding-edge malware technologies affecting systems across the globe.

page 12

ESSENTIAL READING

Martin Overton looks under the covers of the first AVIEN book and reports his findings.

page 17

vbSpam supplement

This month: anti-spam news & events, and Battista Biggio and colleagues present an experimental analysis of the performance of popular spam filter *SpamAssassin* against obfuscated spam images.

COMMENT

'Search engines are free, powerful and efficient tools that can be used to find vulnerabilities and hacked sites.'

Alex Eckelberry
Sunbelt Software, USA

SEARCH ENGINES IN RESEARCH AND VULNERABILITY ASSESSMENT

On 2 October an odd thing happened: state and government websites in California started shutting down. It turned out that the US General Services Administration (GSA) had overreacted somewhat to reports of pornography being hosted on a website for the Transportation Authority of Marin County, and simply pulled the plug on the entire ca.gov domain.

News that the Marin County website was serving porn came as no great surprise to malware researchers. A number of individuals had contacted the owners of the site in mid-September, alerting them to the problem. Unfortunately these alerts were ignored, as they were believed to be 'phishing' attempts.

The primary source of the problem was an apparent DNS hack, which redirected parts of the Marin County website to pornographic sites. These were redirects – the government site itself was not hosting porn. The hack occurred at an outsourced provider, which didn't have the tightest security practices in place.

As many in the research community know, finding these types of hack is trivial work – it can often be a matter of using simple *Google* searches such as *sex porn site:gov*.

Editor: Helen Martin

Technical Consultant: John Hawes

Technical Editor: Morton Swimmer

Consulting Editors:
Nick FitzGerald, *Independent consultant, NZ*
Ian Whalley, *IBM Research, USA*
Richard Ford, *Florida Institute of Technology, USA*
Edward Wilding, *Data Genetics, UK*

Malware researchers are quite familiar with the power of search engines in conducting research. Search engines are free, powerful and efficient tools that can be used to find vulnerabilities and hacked sites on the web, and even in your own organization.

Generally, one sees websites compromised through stolen FTP credentials; unpatched (usually open source) software, including poorly maintained LAMP stacks; the increasing use of collaborative, 'web 2.0'-type software (wikis, tikis, etc.); DNS hacks; poorly written ASP code; sloppy PHP work and SQL hacks.

So-called '*Google* dorks' can be useful in finding compromised and malware-hosting websites. The term was coined by Johnny Long on his website johnny.ihackstuff.com, where he described the practice (which had been around for some time) of using *Google* searches to find 'dorks' – people who expose too much information on the web.

'*Google* dorking' has evolved, and malware researchers continue to fine-tune their searches to find vulnerable websites and malware. Furthermore, by using *Google*'s Alerts feature, one can input a number of different searches and receive alerts whenever a matching site is found – useful in finding newly compromised or rogue sites.

As one example, some broader starter queries might be any of the following: *inurl:traff site:.biz* (or *.info*), *inurl:in.cgi site:.info*, *inurl:klik site:.info*, or *intitle:"index of"* (the last followed by any of a number of terms, such as *"love exe"*, *"jpg exe"*, *porn exe*, *xxx pif*, *"bot exe"* or *"gif exe"* – hence, a final search might be *intitle:"index of" xxx pif -filetype:html -filetype:php -filetype:htm* or, as another example, *intitle:"index of" "bot exe" -filetype:html -filetype:php -filetype:htm*).

Since 'jump pages' are often created with the sole purpose of being indexed on search engines and redirecting visitors to other content, running searches on something like *site:nm.ru* might prove useful. Alternatively, searches can be performed for specific directory structures of frameworks used in malware, such as */stata/index.php*.

Pornography and malware distributors commonly hack into websites for search engine optimization and increased distribution (ironically, *Google*'s work in marking sites as 'unsafe' in search results is likely driving malware and porn distributors to rely increasingly on hacking 'good sites' to perform redirections to their own bad sites). Finding these hacked sites is similarly trivial. One can simply look for any combination of terms, such as *porn*, *free ringtones*, *free casino*, followed by some operators to narrow down the search.

Some knowledge of the language used by the distributors also helps – 'sesso' and 'fottilo', for example, are often used by Italian malware and porn distributors (such as Gromozon). At the time of writing this article, the search *sesso OR gratuito porno OR fottilo site:gov* produces some rather interesting (and sometimes very dangerous) results.

One can continue to experiment by adding different domains and additional operators to the searches. It's common to find plenty of comment spam using these methods, but very often you'll also find compromised websites.

Organizations large and small can use similar searches to find vulnerabilities on their own sites. This holds especially true for larger organizations that work in collaborative environments, such as academic institutions and some governmental organizations. For example, problems with vulnerabilities commonly exist in colleges and universities, where students are often provided with their own websites, academic discourse is encouraged through open source collaborative software, and servers are managed by different groups throughout a campus. It's a recipe for disaster, and that's often exactly what happens – finding hacked university websites is almost trite work. IT administrators could complement their security toolboxes with search engines, seeking inappropriate content on their own domains.

In addition to finding malware on the web, there are numerous (and often hair-raising) searches available that can be used to find vulnerabilities on a site. Queries are limited only by creativity, technical acumen and knowledge of data structures.

Finally, there are distinct differences between search engines. *Yahoo*, *Google* and *Live.com* present similar data, but sometimes one provides clearer results than the other. *Live.com* has the powerful and unique feature of allowing IP searches. Searching for common malware IPs produces profitable results, such as *ip:89.28.13.208*, *ip:89.28.13.213*, and so on.

Some researchers are frustrated by the inability to search within the source of web pages – which, if provided, would open up a mother lode of information, obviating the need to use proprietary spiders. For now, however, one can find plenty of information using simple searches, enabling the research community to find bad things before they get out broadly to the public – and in the process, hopefully making some impact on the safety of the Internet experience for users.

Sunbelt researchers Francesco Benedini and Adam Thomas contributed to this article.

Prevalence Table – September 2007

Virus	Type	Incidents	Reports
W32/Netsky	Worm	1,769,725	35.68%
W32/Mydoom	Worm	901,686	18.18%
W32/Bagle	Worm	704,871	14.21%
W32/Mytob	Worm	451,229	9.10%
W32/Virut	File	382,556	7.71%
W32/Zafi	Worm	105,798	2.13%
W32/Bagz	Worm	96,169	1.94%
W32/Rontokbro	Worm	54,937	1.11%
W32/Stration	Worm	45,484	0.92%
W32/VB	Worm	43,517	0.88%
W32/Lovgate	Worm	36,496	0.74%
W32/Sality	File	35,185	0.71%
W32/Rjump	Worm	34,375	0.69%
W32/Parite	File	33,773	0.68%
W32/Funner	File	23,833	0.48%
VBS/Small	Worm	22,085	0.45%
W32/Klez	Worm	18,853	0.38%
W32/Jeefo	File	17,237	0.35%
W32/Fono	File	14,969	0.30%
W32/SirCam	File	12,751	0.26%
W32/Sohanad	Worm	11,616	0.23%
W32/Small	File	11,278	0.23%
W32/Looked	File	11,096	0.22%
W32/Perlovga	Worm	10,237	0.21%
W32/Hakaglan	Worm	8,878	0.18%
VBS/Butsur	Script	8,680	0.17%
W32/Nuwar	Worm	8,651	0.17%
W32/Tenga	File	8,187	0.17%
W32/Allaple	Worm	6,240	0.13%
W32/Mabutu	Worm	6,122	0.12%
W32/Wukill	Worm	6,075	0.12%
W32/Yaha	File	4,848	0.10%
Others[1]		52,804	0.99%
Total			100%

[1]The Prevalence Table includes a total of 52,804 reports across 170 further viruses. Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

NEWS

E-CRIME UNIT TO GET GOVERNMENT FUNDING?

The UK government has indicated that it may set up a new national police unit dedicated to tackling computer crime. The hint comes as part of the government's response to a report issued earlier this year by the Science and Technology Committee of the House of Lords, which strongly advised it to take steps to improve the policing of the Internet.

Specifically, the House of Lords report urged the Home Office to provide the necessary funds to kick-start the establishment of the Police Central e-crime Unit (PCeU) – a project driven by members of the Metropolitan Police to create a central coordination point for e-crime reporting. The government responded by saying that it will 'consider the proposals to create a law enforcement unit to tackle crimes involving computers.'

The e-crime unit, run by the Metropolitan Police, would act as a central coordination point for the e-crime divisions of the UK's local police forces, providing training for officers, collating e-crime reports and liaising with the Serious Organised Crime Agency (SOCA). Its remit would cover similar areas to those of the now defunct National High Tech Crime Unit (NHTCU), which was disbanded and absorbed into SOCA in April 2006.

The project has already received financial support from the National Policing Improvement Agency (NPIA), but it has yet to be guaranteed central government funding and will need to turn to private sector backers if the government fails to allocate the necessary funds.

The government's response to the House of Lords report can be read in full at http://www.official-documents.gov.uk/document/cm72/7234/7234.pdf.

VB100 NETWARE UPDATE

VB regrets that some erroneous results were recorded for *Symantec AntiVirus 10* in last month's comparative review on *Novell NetWare 6.5* (see http://www.virusbtn.com/pdf/magazine/2007/200710_VB100.pdf). The product was stated to have missed five samples from the polymorphic set – however it has since been discovered that, as a result of file-copying errors, several corrupted samples were included in the test set used to test the *Symantec* product. After removing the corrupted samples and retesting the product, *Symantec AntiVirus 10* was found to detect all files in the set, giving it a faultless 100% detection rating across all test sets.

Virus Bulletin apologises both to readers and to *Symantec* for this error. Measures will be introduced into the VB100 testing process to ensure test sets are kept intact for all tests in future. No other vendors were affected.

LETTER

APPLICATION WHITELISTING

Multi-layered defence is a very good thing, but only if the applied layers are more or less independent of each other. Are application whitelisting and AV scanning statistically independent?

I will resist the temptation to discuss whether different AV scanners are independent nowadays [1] – and only say that they are not as independent as I would like them to be.

Let us look at the process of how a program becomes whitelisted. The classification can be made for two reasons: either because the software is trusted, or because it has produced negative AV scan results.

In the case of 'trusted' software developers, the program inherits the trust of the company that has developed it, on the assumption that whatever they release is clean. Is this because we know with certainty that the development process is free from possible infection? No – in the past even major software developers have released infected executables. But having learned from these experiences, an intensive AV scan (with multiple-scanner systems) is now incorporated by these developers into the release process of their products. So the trust is based on the fact that the release software goes through virus scanning.

In the case of untrusted developers the file is whitelisted if it comes up negative against an extensive virus search.

Either way, the decision to whitelist a program comes from a negative virus scan result at some point in the process. In short, whitelisting is currently nothing more than (admittedly careful and extensive) inverted blacklisting by AV software.

In order to become a really valuable component of a layered defence, whitelisting must be independent of AV scanning. This would require a thorough analysis of indexed files. The problem is not only that there are several orders of magnitude more files to be whitelisted than blacklisted. In addition, a lot more analysis is required to declare that a file is clean than to declare that it is malicious. For a malicious file, the analysis can be stopped at the first sign of malicious behaviour (during dynamic analysis) or at the first malicious procedure encountered in the code (during static analysis). For a clean file, all procedures have to be checked to be certain that they are incapable of malicious act. In short, only a full, detailed analysis can verify the decision to whitelist a file – a lot more resources need to be dedicated to a good whitelisting database than to a virus database. It is resource-intensive and time-consuming, but not impossible.

Gabor Szappanos
VirusBuster, Hungary

[1] Canja, V. Exploiting the testing system. International Antivirus Testing Workshop 2007, Reykjavik.

ANALYSIS

SPAM FROM THE KERNEL

Kimmo Kasslin
F-Secure, Malaysia

Elia Florio
Symantec, Ireland

In December 2006 one of the authors of this article concluded his research paper on kernel malware with the following paragraph [1]:

> 'This paper has shown the basic techniques that kernel malware is using to do their job. Their main role has been to perform some specific tasks for the main user-mode component. However, the scene is changing. There has been lots of interest in various research groups to investigate for the possibilities to do more complex tasks directly from kernel. The next big thing is going to be the network side. This year we have already seen presentations talking about how backdoors can be implemented directly from kernel mode using only the NDIS layer and custom TCP/IP stack. We have also seen a presentation about bypassing personal firewalls from kernel-mode.'

Regrettably, the prediction became true in June 2007 when the authors started to analyse a malicious kernel-mode driver consisting of large amounts of code. After deeper analysis it became evident that there were no signs of user-mode injection, meaning that all of the code was being executed at ring 0. Since then the malware family has been known as Trojan.Srizbi (or as Rootkit:W32/Agent.EA).

ORIGIN OF TROJAN.SRIZBI

In June 2007 several AV companies raised an alert due to a large-scale web-based attack being carried out with drive-by exploits. A malicious tag was injected into the homepages of many legitimate Italian and Russian websites. Attackers had injected an IFRAME to redirect visitors to a website running a multi-staged exploit kit, nowadays best known as 'MPack' [2]. The specific MPack installation used for this attack was designed to download several trojans from the remote server 81.95.146.150, hosted by the ambiguous 'Russian Business Network'.

One of the nasty creatures to spread from this server was Trojan.Srizbi, a sophisticated piece of malware contained within 150Kb of kernel mode code. This was the first public revelation of Srizbi, but further research tracked older samples back to April 2007. The first observed filename was 'windbg48.sys', followed by 'symavc32.sys' more recently. Other pseudo random names of this threat always take the format [CHARS][2DIGITS] (e.g. Cmjg57.sys, Wdml36.sys, Fln51.sys, etc.).

The Srizbi driver is installed by a dropper component that is packed with UPX and protected with a layer of scrambled code. This layer uses custom spaghetti code obfuscation with CALL/JMP mixed with junk opcodes. It resembles the Rustock.B polymorphic packer, but it is more advanced. The dropper code makes API calls by using PUSH/RET sequences, and parameters are pushed into the stack with MOV [ESP] and other complex indirect loading instructions, making standard static analysis almost impossible (see Figure 1).

Reverse engineering of this piece of code is not straightforward and may require the assistance of custom tracing tools. However, it is possible to spot two principal decryption routines in the code:

(1) decryption with NOT BYTE

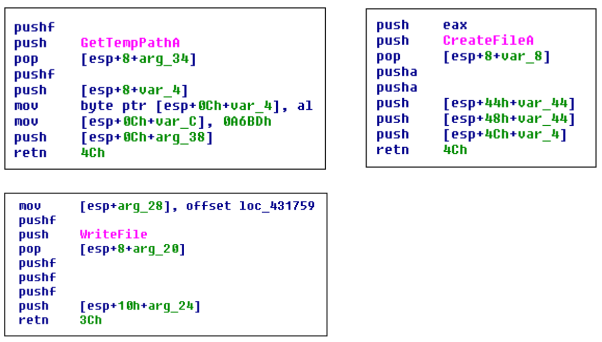
(2) decryption with BYTE XOR 0xB0

Decryption routine (1) is identical to the one inside the driver and is used to decrypt all text strings; the XOR decryption (2) is used to decrypt the embedded driver. The Srizbi installer drops a copy of the driver into the '%SystemRoot%\System32\drivers' folder and installs it as a service via OpenSCManager and CreateService. Next, the service is started and the installer self-deletes.

Another of Srizbi's challenges is the use of CRC values instead of basic string compares when it searches for names or resolves imported API functions dynamically. In some cases the only viable solution to find which string is being searched by Srizbi is to brute force CRCs over a set of possible strings.

FULL-KERNEL MALWARE

We can say with some certainty that Trojan.Srizbi represents an important milestone in the evolution of malware utilizing kernel-mode techniques. It is the first complex *full-kernel malware*[1] to have features such as file



```
pushf
push    GetTempPathA
pop     [esp+8+arg_34]
pushf
push    [esp+8+var_4]
mov     byte ptr [esp+0Ch+var_4], al
mov     [esp+0Ch+var_C], 0A6BDh
push    [esp+0Ch+arg_38]
retn    4Ch
```

```
push    eax
push    CreateFileA
pop     [esp+8+var_8]
pusha
pusha
push    [esp+44h+var_44]
push    [esp+48h+var_44]
push    [esp+4Ch+var_4]
retn    4Ch
```

```
mov     [esp+arg_28], offset loc_431759
pushf
push    WriteFile
pop     [esp+8+arg_20]
pushf
pushf
pushf
push    [esp+10h+arg_24]
retn    3Ch
```

Figure 1: Examples of scrambled API calls used by the Srizbi packer.

and registry hiding, bypassing of memory hooks, and low-level NDIS hooks with a private TCP/IP stack. The latter is utilized to implement a fully blown spam client with an HTTP-based command and control (C&C) infrastructure – all directly from ring 0.

The fact that Trojan.Srizbi is fully implemented in kernel mode makes it very powerful. It can activate very early during the boot process, allowing it to make its system modifications before most security products even have a chance to load. Also, many security products do not consider activities initiated by kernel-mode code to be suspicious or malicious since the kernel should be trusted.

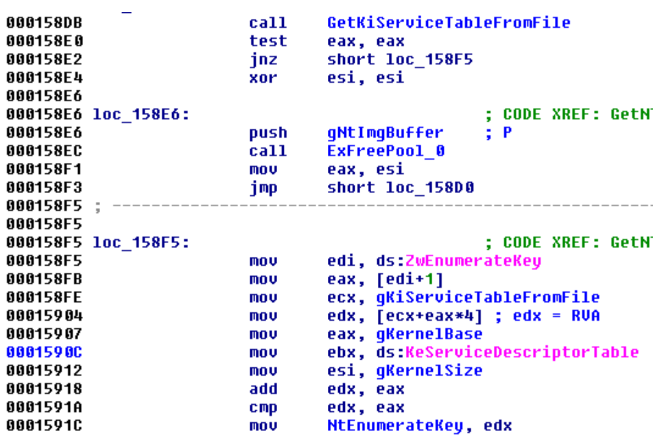
A feature that makes Srizbi unique is the fact that it disables and removes other competitor rootkits from the infected system very effectively. The trend of malware gangs fighting each other is becoming more intense nowadays since every unprotected machine is likely to be targeted by multiple pieces of malware – but in the end, there can be only one! Only the strongest and most sophisticated malware will survive.

Srizbi tries to accomplish this goal in a unique way: it rebuilds a clean copy of KiServiceTable by reading NTOSKRNL.EXE directly from disk and then it retrieves all the original (hook-free) function pointers of the required file and registry API functions. With this technique Srizbi is able to bypass most rootkits (and security products) present on the machine and can safely enumerate any files and registry keys.

We also found that Srizbi contains a generic driver removal routine based on CRC values. The routine enumerates services sub keys from following key:

```
\REGISTRY\MACHINE\SYSTEM\CurrentControlSet\Services\
[SRV_NAME]
```

If the CRC of [SRV_NAME] matches one of the following hardcoded CRC values, then the trojan unloads the driver



```
000158DB                call    GetKiServiceTableFromFile
000158E0                test    eax, eax
000158E2                jnz     short loc_158F5
000158E4                xor     esi, esi
000158E6
000158E6 loc_158E6:                                 ; CODE XREF: GetN
000158E6                push    gNtImgBuffer        ; P
000158EC                call    ExFreePool_0
000158F1                mov     eax, esi
000158F3                jmp     short loc_158D0
000158F5 ; ---------------------------------------------------------------
000158F5
000158F5 loc_158F5:                                 ; CODE XREF: GetN
000158F5                mov     edi, ds:ZwEnumerateKey
000158FB                mov     eax, [edi+1]
000158FE                mov     ecx, gKiServiceTableFromFile
00015904                mov     edx, [ecx+eax*4] ; edx = RVA
00015907                mov     eax, gKernelBase
0001590C                mov     ebx, ds:KeServiceDescriptorTable
00015912                mov     esi, gKernelSize
00015918                add     edx, eax
0001591A                cmp     edx, eax
0001591C                mov     NtEnumerateKey, edx
```

Figure 2: Srizbi rebuilds its own Service Table from disk.

with ZwUnloadDriver then deletes the launch point of the service by removing the related registry key and its sub keys.

CRC value	Service name
0xe0e5a117	runtime
0x4c4f27cc	runtime2
0xbf36b345	xpdx
0x949b30b3	lzx32

Table 1: CRC values and service names targeted by Srizbi's driver removal routine.

The services in Table 1 are quite (in)famous within the AV industry, since they are used by two of today's most common rootkit malware families: Backdoor.Rustock.B (lzx32, xpdx) and Trojan.Pandex (runtime, runtime2). Srizbi also contains the text strings of some other malware, 'wincom32.sys' and 'ntio256.sys', however they are not referenced and never used in the actual code. More evidence of the ongoing war between the Srizbi gang and other malware authors was reported by researchers at *Arbor Networks* [3], who noticed StormWorm bots running DDoS attacks against Srizbi domains.

NDIS HIJACKING

From a technical point of view the most interesting part of Trojan.Srizbi is its network layer implementation. Its sole purpose is to bypass personal firewalls and other security products that monitor incoming and outgoing network packets. The implication is that the infected machine will be able to communicate with the command and control server and send thousands of spam emails even if the firewall is set to the 'block-all-traffic' mode.

Windows networking architecture is divided into different layers where the ones most commonly used by today's firewalls are TDI and NDIS. The NDIS layer abstracts the network hardware from network drivers and is the lowest layer available for third-party network drivers. This makes it the obvious choice for modern firewalls since the lower they operate the harder it is to bypass them.

Usually NDIS hooking firewalls install their triggers to the following locations:

- NDIS library functions
- NDIS_PROTOCOL_BLOCK structure handler function pointers
- NDIS_OPEN_BLOCK structure handler function pointers

The actual triggers controlling inbound and outbound traffic are implemented by replacing several function pointers from inside the two internal NDIS structures that the NDIS layer uses for sending packets to and receiving packets from the driver controlling the hardware. The following are some commonly hooked handler functions:

- NDIS_OPEN_BLOCK->SendHandler
- NDIS_OPEN_BLOCK->SendPacketsHandler
- NDIS_OPEN_BLOCK->ReceiveHandler
- NDIS_OPEN_BLOCK->ReceivePacketHandler

The firewall driver normally installs inline hooks into four functions exported by ndis.sys that allow it to hook the required NDIS handler functions for any newly registered protocol driver. Otherwise the new protocol driver would be able to send and receive traffic without the firewall seeing it. The hooked functions are:

- NdisRegisterProtocol
- NdisDeregisterProtocol
- NdisOpenAdapter
- NdisCloseAdapter

Now that we have had a brief introduction to how modern firewalls are able to filter the traffic, let's continue by looking at what makes it possible for Trojan.Srizbi to bypass them completely.

First, Trojan.Srizbi finds a network adapter that it can use to communicate with the Internet. It does this by finding a suitable interface from the TCP/IP driver's registry settings. It calls an undocumented LookupRoute function exported by tcpip.sys which will give it the IP address of the default gateway. Then it enumerates all sub keys under the key:

\REGISTRY\MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces

It uses the default gateway address to find the matching interface, or if there are no matches it will take the one that has proper IP settings defined. Next, it enumerates all sub keys under the key:

\REGISTRY\MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Adapters

Finally, it selects the adapter whose IpConfig value matches the previously found interface.

Trojan.Srizbi gets access to the NDIS structures by installing a dummy protocol temporarily. It first registers a new protocol with a random name by calling NdisRegisterProtocol. Then it binds the protocol to the previously selected adapter by calling NdisOpenAdapter. As a result, the temporary protocol's NDIS_OPEN_BLOCK handler functions will be set up by the underlying system.

As we have already mentioned these two functions are commonly hooked by the firewall. Trojan.Srizbi solves this problem by using its private and hook-free version of ndis.sys. During initialization it loads the ndis.sys file into memory, resolves its imports and finally relocates the new module into the base address of the original ndis.sys module. This means that the private module will still be using e.g. the same global variables as the original ndis.sys module.

The following is the disassembly of the private and original NdisRegisterProtocol functions:

```
kd> u poi(Yol33!NdisRegisterProtocol)
816b917d 8bff           mov     edi,edi
816b917f 55             push    ebp
816b9180 8bec           mov     ebp,esp
816b9182 51             push    ecx
816b9183 53             push    ebx
816b9184 56             push    esi
816b9185 57             push    edi
816b9186 b938d16ff9     mov     ecx,offset
NDIS!ndisPkgs+0x20 (f96fd138)

kd> u ndis!NdisRegisterProtocol
NDIS!NdisRegisterProtocol:
f96ff17d e9b2d75200     jmp     fwdrv+0x934 (f9c2c934)
f96ff182 51             push    ecx
f96ff183 53             push    ebx
f96ff184 56             push    esi
f96ff185 57             push    edi
f96ff186 b938d16ff9     mov     ecx,offset
NDIS!ndisPkgs+0x20 (f96fd138)
```

With the help of the dummy protocol the driver now has a handle to the registered protocol that is bound to the underlying adapter. The handle is returned by NdisRegisterProtocol's second argument called NdisProtocolHandle, which is defined as PNDIS_HANDLE. This is actually a pointer to the dummy protocol's NDIS_PROTOCOL_BLOCK. By using the information stored in this structure the malware is able to find the adapter's NDIS_MINIPORT_BLOCK structure that is part of the lowest layers of NDIS. It then searches for other protocols that are bound to the same adapter, and takes the first one to match any of the following protocols:

1. TCPIP
2. PSCHED
3. TCPIP_WANARP

The way this whole process works is illustrated in Figure 3.

Now that the driver has access to the NDIS structures of an existing protocol it will replace a bunch of handler functions with its own. In addition, it fetches the addresses of certain handler functions that it will use itself. After it has finished it will just uninstall the dummy protocol.

Trojan.Srizbi uses its NDIS hooks and the original handler functions together with its own TCP/IP stack to send and receive packets.

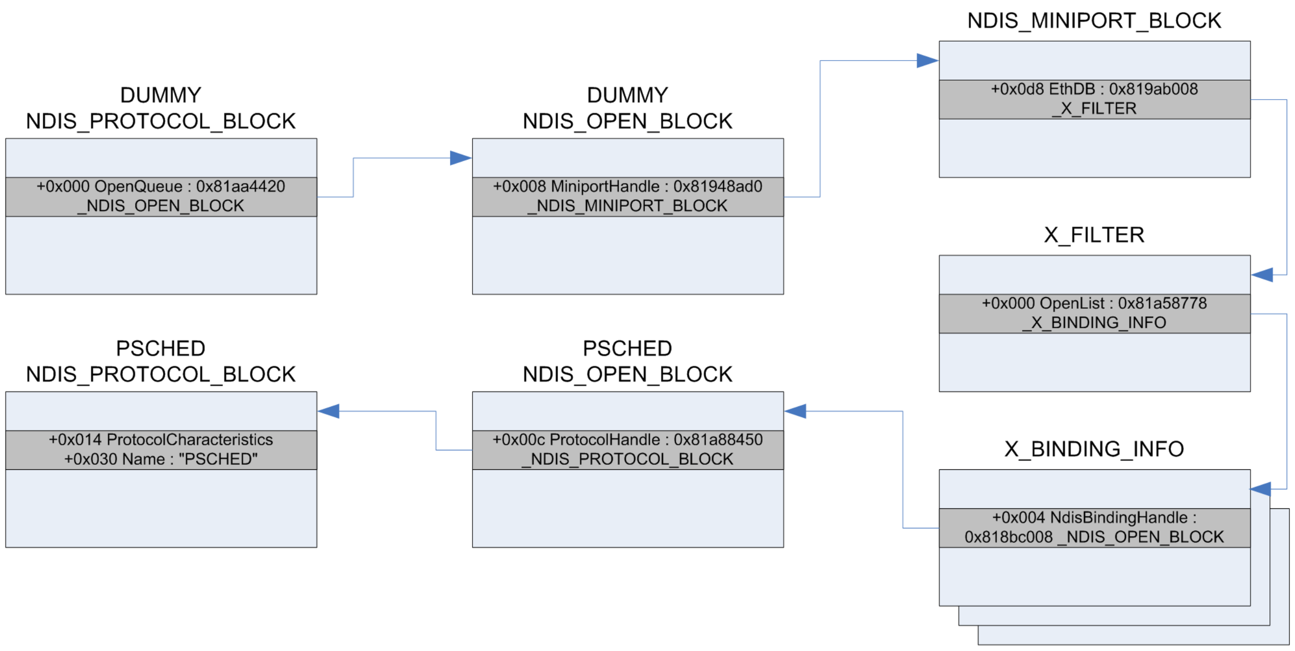

DUMMY NDIS_PROTOCOL_BLOCK
+0x000 OpenQueue : 0x81aa4420 _NDIS_OPEN_BLOCK
DUMMY NDIS_OPEN_BLOCK
+0x008 MiniportHandle : 0x81948ad0 _NDIS_MINIPORT_BLOCK
NDIS_MINIPORT_BLOCK
+0x0d8 EthDB : 0x819ab008 _X_FILTER
X_FILTER
+0x000 OpenList : 0x81a58778 _X_BINDING_INFO
X_BINDING_INFO
+0x004 NdisBindingHandle : 0x818bc008 _NDIS_OPEN_BLOCK
PSCHED NDIS_OPEN_BLOCK
+0x00c ProtocolHandle : 0x81a88450 _NDIS_PROTOCOL_BLOCK
PSCHED NDIS_PROTOCOL_BLOCK
+0x014 ProtocolCharacteristics
+0x030 Name : "PSCHED"

Figure 3: Trojan.Srizbi uses the 'dummy protocol' approach to find the NDIS structures that it hooks.

To send packets it uses the following handler function:

- NDIS_MINIPORT_BLOCK->SendPacketsHandler

To get a notification after the send operation has completed it uses the following hook:

- NDIS_OPEN_BLOCK->SendCompleteHandler

To receive packets it uses the following hooks:

- NDIS_OPEN_BLOCK->ReceiveHandler
- NDIS_OPEN_BLOCK->ReceivePacketHandler

Since the malware is using its own TCP/IP stack it has somehow to identify which of the received packets should be passed to its private stack instead of the stack used by the system. One solution to this problem would be to use its own IP and MAC addresses since their combination should be unique on most physical machines [4].

However, Trojan.Srizbi uses a different approach. When it sends packets it always uses a source port that is higher than 32,000. For every packet received it checks whether it is a TCP packet, whether its destination IP equals the client's address, and whether the destination port is larger than 32,000. If all of these are true then it forwards the packet to its private stack. To make sure that no other application (using ephemeral ports) sends packets using its reserved source port range it sets the MaxUserPort registry value to 31,999, which is defined under the following key:

\REGISTRY\MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters

STEALTH SPAM

Srizbi (like Rustock) is a spam bot. A sophisticated, stealthy and powerful spam machine. Once the trojan takes over NDIS networking, it contacts C&C servers on TCP port 4099 and retrieves spam instructions and configurations. The spam backdoor has a large set of commands that allow the botmaster to define a lot of parameters. A deep analysis of the C&C protocol is beyond the scope of this document, but we can briefly summarize how it works. Each spam session starts with the download of a ZIP package containing the files shown in Table 2.

File	**Content**
config	Configuration file with all spam parameters (e.g. task_owner, max_mails, max_sim conn, pipeline, subj, etc.)
message	Text/HTML body of the spam message
mlist	List of recipients
000_data22	List of mail domains
001_ncommall	List of names/surnames
002_senderna	List of names/surnames
003_sendersu	List of names/surnames
mxdata	MX records of the domains

Table 2: Configuration files included in the package provided by the Srizbi C&C server.

Srizbi also has an advanced feature that allows the trojan eventually to bypass some badly configured honeypot machines. The trojan does not trust the locally configured DNS server of the infected machine and instead receives all the necessary DNS information as part of the ZIP package. For example, if Srizbi needs to resolve the 'smtp.acme.org' server to send spam, it will receive in the package the necessary MX record info for the 'acme.org' domain. Any honeypot that simply blocks/redirects DNS resolutions to prevent threats from spamming will be bypassed by Srizbi because it has a kind of private DNS server over the C&C channel. Srizbi can send image spam in HTML format using English and also Cyrillic Unicode character sets.

CONCLUSION

Despite all the advanced features implemented for spam and networking, the major weakness of Srizbi is its hiding technique. The rootkit attempts to hide itself by placing the following kernel hooks:

- Inline hook: NtOpenKey, NtEnumerateKey
- \FileSystem\Ntfs driver: IRP_MJ_CREATE, IRP_MJ_DIRECTORY_CONTROL;

These hooks effectively hide the malware's driver file and registry keys, but currently they can easily be detected by common rootkit detectors and can effectively be bypassed to remove the threat from the infected system.

REFERENCES

[1] Kasslin, K. Kernel malware: the attack from within. 2006. http://www.f-secure.com/weblog/archives/kasslin_AVAR2006_KernelMalware_paper.pdf.

[2] Symantec. MPack – The Movie. http://www.symantec.com/enterprise/security_response/weblog/2007/06/mpack_the_movie.html.

[3] McPherson, D. When spambots attack – each other! http://asert.arbornetworks.com/2007/07/when-spambots-attack-each-other/.

[4] Hoglund, G.; Butler, J. Rootkits: subverting the Windows kernel. 2005. Upper Saddle River, NJ. Addison-Wesley Professional. 324 pages. ISBN 0-321-29431-9.

FOOTNOTES

[1]Malicious code that executes all its code in ring 0. After it is installed into the system by a dropper component (user-mode code or kernel-mode exploit) it will be able to operate by itself.

FEATURE 1

ANONYMOUS PROXIES: THE THREAT TO CORPORATE SECURITY ENFORCEMENT

Rony Michaely
Aladdin, Israel

Anonymous proxies emerged as a result of the 'fighting Internet censorship' movement and have grown to become one of the leading security threats to corporations, educational institutions and other organizations, as well as end-users worldwide.

The past year has witnessed a dramatic increase in the number of anonymous proxy services on offer. The phenomenon started in 2002 with a few dozen sites offering users anonymous access to Internet resources, and now over 100,000 registered websites and an estimated 300,000 private, home-based websites offer anonymity services.

The main reason for this dramatic increase is that there has been an increase in the number of users desiring such sevices. Many business-minded individuals have seized the opportunity to make money through charging users a monthly fee for anonymity services. Another reason for the increase in these services relates to technology. Software running on proxy anonymizer sites has become open source, making web-based proxies available to anyone who wants to access them. This new open-source approach gives even relatively non-technical users the ability to create anonymous proxies on the fly. These proxies are then placed on newly created or home-based websites, bypassing Internet filters.

HOW ANONYMOUS PROXIES WORK

Anonymous proxies are probably the most popular and effective way for users to bypass Internet filters. Appearing as an unblocked web page, a proxy anonymizer site allows a user to enter any URL into a form. When the form is submitted, the proxy server retrieves the web page even if it is blocked by the organization's Internet filter.

Access to open-source anonymous proxies is based on two main methods:

- CGI-proxy. Through a CGI Script, users can retrieve any resource that is accessible from the server on which it runs. When an HTML resource is retrieved, it is modified so that all links in it refer back to the same proxy, including images and form submissions. Configurable options include text-only support, SSL support, selective cookie and script removal, simple ad filtering, access restriction by server, and custom encoding of target URLs and cookies.

- PHP-proxy. A web HTTP proxy programmed in PHP can easily be installed on any PHP-enabled web server. It allows users to browse through the web server itself as a proxy for bypassing firewalls and other content filter restrictions. PHP-proxy uses a web interface that is very similar to the popular CGI-proxy.

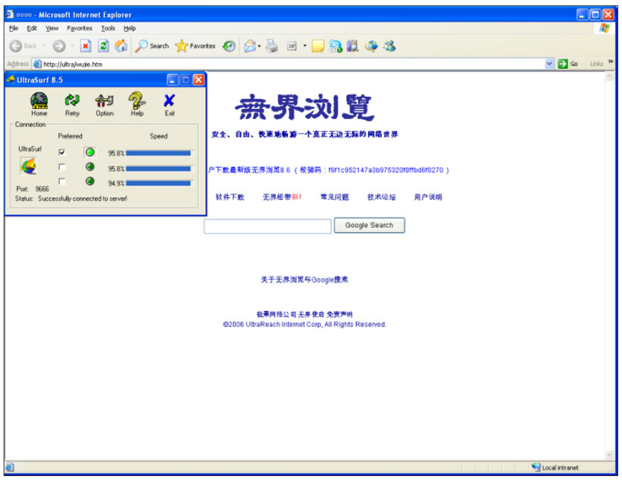

Microsoft Internet Explorer
UltraSurf 8.5
Home
Retry
Option
Help
Exit
Connection
Preferred
Speed
Port: 9666
Status: Successfully connected to server!
无界浏覽
Google Search
©2006 UltraReach Internet Corp, All Rights Reserved

Figure 1: Encapsulation of HTTP traffic into an SSL tunnel.

RISKS POSED BY ANONYMOUS PROXIES

Anonymous proxies pose a range of risks:

- In schools they allow students to access sites prohibited by their school's Internet policy, which may be inappropriate and potentially harmful.
- They expose organizations to drive-by spyware, viruses and trojans.
- They expose users to identity theft, pharming and phishing attacks.
- They expose organizations to information theft.
- They provide anonymity for abusers of corporate resources (e.g. workers using company systems for illegal activities, posting inappropriate content etc.)
- They prevent web filters from monitoring users' online activities.

INAPPROPRIATE INTERNET USAGE HITS THE HEADLINES

The CIPA (Children's Internet Protection Act) is a federal law that was enacted by the American Congress in December 2000 to address concerns about access to offensive content via the Internet in schools and libraries. CIPA clearly requires schools and libraries to operate a 'technology protection measure' with respect to any of its computers with Internet access that 'protects against access through such computers to visual depictions that are obscene, child pornography, or harmful to minors'. The following excerpts illustrate the growing threat of anonymous proxies from the aspect of enforcing Internet usage policy in corporations and educational institutions:

Teacher Allegedly Viewed Porn at Library (7 July 2007)

'Tulsa County prosecutors charged William Lee Hunter Jr. on Thursday with procuring or possessing child pornography at the Central Library. Tulsa Public Schools records show that Hunter taught during the 2006–07 school year at Springdale Elementary School.'

Source: www.tulsaworld.com

Worker Fired for Viewing Porn on Job (5 July 2007)

'A state employee who policed Internet usage by other state workers has been fired for viewing pornography on his own office computer. Thomas Rice of Grimes was fired in May from the Iowa Public Employees Retirement System, where he worked as a top-level information technology specialist. Rice's supervisors allege that over a nine-day period in March he viewed dozens, if not hundreds, of pornographic images and movies throughout the workday.'

Source: desmoinesregister.com

THE MALICIOUS ASPECT

Analysis of publicly available anonymous proxies found that 5% of these servers contained malicious content. Server directories were found to contain infected files including trojans, script viruses and exploits, spyware and adware.

Vulnerability analysis carried out by *Aladdin CSRT* on 1,000 registered anonymous proxy websites showed that 70% of these sites were vulnerable to remote code execution and cross-site scripting attacks (see Figure 2).

Vulnerabilities found on anonymous proxy sites included:

- Cross-site scripting (high severity)
- PHP Zend_Hash_Del_Key_Or_Index (high severity)
- PHP HTML entity encoder heap overflow (high severity)
- CRLF injection/HTTP response splitting (high severity)
- SQL injection (high severity)
- PHP version older than 4.4.1 (high severity)
- Apache chunked encoding exploit (high severity)
- OpenSSL ASN.1 deallocation (high severity)

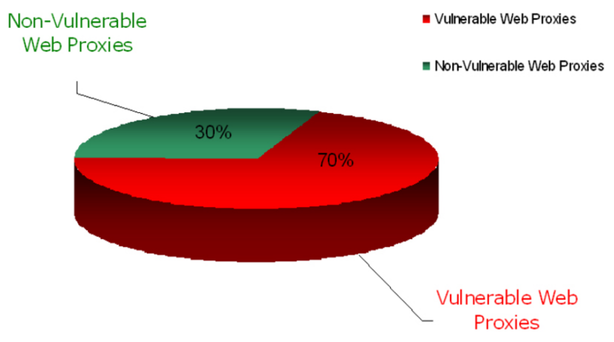
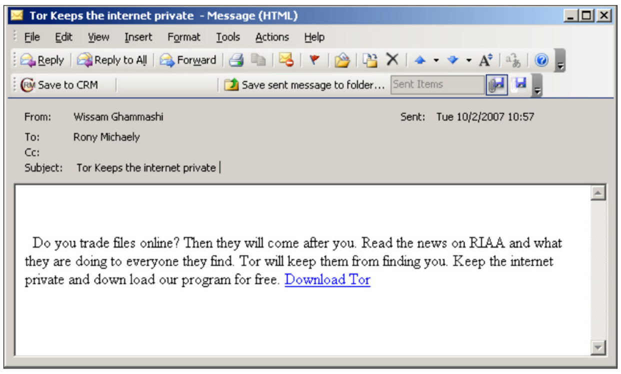


Figure 2: Percentage of vulnerable anonymous proxies where 1,000 sites were tested.

- SSL PCT handshake overflow (high severity)
- PHP version older than 4.3.8 (medium severity)
- Apache 2.x version older than 2.0.55 (medium severity)
- Apache error log escape sequence injection (medium severity)
- Apache Mod_Rewrite Off-By-One buffer overflow (medium severity)
- PHP unspecified remote arbitrary file upload (medium severity)
- Remote directory traversal (medium severity)

These vulnerabilities can potentially be exploited for malicious purposes including: remote code execution, cross-site scripting, denial of service attacks, privilege escalation and poisoning of the web cache.

The latest variants of the Storm worm launched a new kind of social-engineering attack, using spam to urge users to use online anonymity system *Tor* for their communications. The message contained a link to download a malicious version of *Tor* (see Figure 3).


Tor Keeps the internet private - Message (HTML)
File Edit View Insert Format Tools Actions Help
Reply Reply to All Forward
Save to CRM Save sent message to folder… Sent Items

From: Wissam Ghammashi Sent: Tue 10/2/2007 10:57
To: Rony Michaely
Cc:
Subject: Tor Keeps the internet private

Do you trade files online? Then they will come after you. Read the news on RIAA and what they are doing to everyone they find. Tor will keep them from finding you. Keep the internet private and down load our program for free. Download Tor

Figure 3: The latest variants of the Storm worm used spam to convince users of the necessity of using Tor for their communications.

WHY DO MOST CONTENT-FILTERING PRODUCTS FAIL TO HANDLE THIS THREAT?

Although most Internet-filtering solutions include an ‘anonymous proxy’ or ‘proxy avoidance’ category in their databases, they actually fail to block access to web-based proxies due to their list-based approach. List-based products cannot keep up with the increasing number of new proxy sites. The fact that users can easily install anonymous proxies on their private computers makes it even harder. The most crucial element that makes anonymous proxies a leading security threat and problematic for security products is the SSL support offered by many of these servers. Over 30% of the websites that offer anonymous surfing allow SSL connection.

THE BATTLE AGAINST ANONYMOUS PROXIES

There are several things that can be done to block access to anonymous proxies within organizations:

- Analysing form methods and meta tags will prevent access to an estimated 40% of these websites.
- Pattern-based detection and HTTP header analysis will catch requests for anonymous proxies on the fly, providing organizations with protection against circumvention and anonymity techniques.
- Only 5% of the SSL-enabled anonymous proxies we analysed provided a valid certificate. All others presented expired, self-signed, mismatched or otherwise doubtful credentials. Validating the SSL certificate and assuring a trusted certificate issuer will prevent access to 95% of these SSL-enabled websites.
- Many URL-filtering products contain an ‘uncategorized’ filter (sites that are not listed by the product). Use of this filter can prevent access to anonymous proxies installed on home computers.

CONCLUSION

The future may see a serious threat as a result of the continued growth of malicious anonymous proxies. The popularity of anonymous proxies is rising rapidly and the number of websites offering anonymous proxy services is increasing dramatically, bringing with it a growing concern in the form of high severity vulnerabilities on most of these sites. Phishing and social-engineering-based attacks aiming to lure users to use or install anonymous proxy services will increase exponentially. Unfortunately, relying on list-based and reactive security systems and continually chasing updates will prove increasingly unreliable.

FEATURE 2

MALWARE STORMS: A GLOBAL CLIMATE CHANGE

Pierre-Marc Bureau and Andrew Lee
ESET, Canada and UK

Since late 2006, the family of W32/Nuwar malware threats, often called the 'Storm worm', have been slowly but surely taking the Internet, if you'll excuse the pun, by storm.

In this article we examine the evolution of Storm from its rather humble beginnings as a minor malware threat into what is probably one of the most bleeding-edge malware technologies currently affecting systems across the globe. We will focus particularly on the ways in which the malware authors have attempted to combine advances in social engineering techniques with changes in behaviour (and particularly network communication) to avoid detection by anti-virus systems. These changes are significant for their impact on anti-malware detection and perhaps more significantly on intrusion detection and prevention systems (IDS and IPS).

For some time, malware researchers (including us) have been predicting the move of botnets away from centralized command and control servers towards decentralized control models using peer-to-peer technology [1]. The Storm worm (also known as Zhelatin, Nuwar, Peacomm, Tibs, Peed and Fuclip) was one of the first bots to be seen using peer-to-peer communications. The bots were now swarming.

It has become increasingly clear that the authors of the Storm family are not simply hobbyists, but rather are investing seriously in their creation, actively seeking to increase its capabilities and to protect it as far as possible from detection.

A STORM BY ANY OTHER NAME...

Storm gets its name from one of its early spam/seeding runs, which was seen in early 2007 just after the severe storm Kyrill that ravaged large parts of Europe. As is typical with many email worms and seeding runs, the messages in the initial Storm seeding runs used a variety of topics related to current news to entice recipients to open the email and execute the attachment. One of the early messages claimed to contain a video of the execution of Saddam Hussein in Iraq. The message came with a file called 'Full Clip.exe' which, when executed, would infect the PC with the latest variant of the Storm worm. One of Storm's common aliases, 'Fuclip', was derived from this seeding run. Earlier subjects involved claims of imminent nuclear war (from which *ESET* took the Nuwar alias), the death of Cuban president Fidel Castro and US aircraft being shot down by missiles.

Interestingly, the Storm worm has changed its seeding methods several times, from initially using the tried and tested 'enticing' email subjects as described above, through using electronic greetings cards (a move that had a huge impact on the revenues of legitimate e-card retailers), via websites containing exploit code, to the most current form – websites that do not contain exploits, but which offer 'free' downloadable content, such as sport scores and computer games. The common factor in all these social engineering techniques is that the malicious files or websites are advertised through emails.

A chronological breakdown of the different social engineering methods used by Storm [2] is shown in Figure 1.

Type	Period
Dramatic/current news	December 2006 – May 2007
Electronic greetings cards	June – August 2007
Electronic postcards	August 2007
Technical support (patch or VPN connector)	August 2007 (one day only)
Beta test program	August 2007 (one day only)
Video	August – September 2007
'Fake' user credentials to sites	August – September 2007
Labor day	September 2007 (one day only)
Privacy software (TOR)	September 2007 (one day only)
NFL season	September 2007
Arcade games download	September – October 2007

Figure 1: Evolution of Storm's social engineering.

Drawing on different techniques of social engineering is probably intended to hit as many 'hot-buttons' as possible. By using techniques from classic email worms, phishing and so on, and covering a wide spread, the worm variants are more likely eventually to be successful. The use of such techniques can also deflect attention, because the threat becomes just another one of the many, rather than being noticed immediately as another Storm worm.

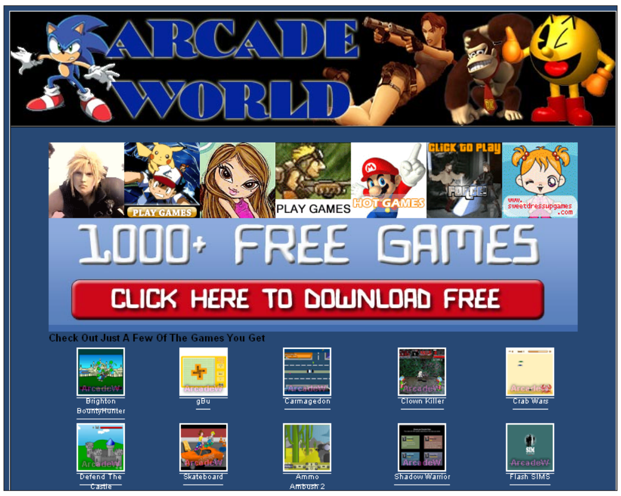


Figure 2: Storm's 'Arcade World' site.

Of course, many businesses filter out executable attachments in email. This may have forced the change seen in seeding methods to those in which only a link was included in the email – which required more sophisticated social engineering to tempt the user. Some of these links directed the user to websites containing code that would exploit vulnerabilities in popular browsers or applications, but as time progressed (and as patches for those vulnerabilities became available) a switch was made to more legitimate-looking sites.

One of the most recent versions of the Storm seeding mails offers arcade games for download (see Figure 2). Of course, all the links on the site simply download ArcadeWorld.exe – a variant of the Storm worm. No doubt by the time this article is published, things will have moved on again.

A GATHERING STORM

Once it's on a system, the biggest strength of the Storm worm is its communication mechanism, which allows its authors to monitor the performance of their botnet closely and to update the code base rapidly. Storm uses its communication for the following purposes:

1. Software update: updated components are delivered frequently.
2. Status update: nodes report their status and uptime to the controller.
3. Payload activation: 'updates' are delivered with instructions for spam runs or DDoS attacks.

During the first six months of its existence, the Storm worm used the Overnet peer-to-peer network protocol to exchange information about the location of update files on the Internet.

To bootstrap the connection to the Overnet network, a node needs to establish connections to a certain number of peers on the network. It does this using a file containing the IP addresses of a number of known peers. When a new node connects, it sends its information to these peers which, in turn, reply (assuming they are online) with a list of other peers to which the new node can connect. In most cases, a newly connected node needs to establish contact with a few hundred peers before it can begin searching for information.

Systems infected with the Storm worm would perform the following steps to connect to the network and find a new version of their binaries:

1. Contact peers that are present in the peer list file contained within the Storm binary itself (Overnet bootstrap process) – this is a list of 260 initial peers.
2. Connect with other peers on the network using information supplied by peers contacted in step 1.

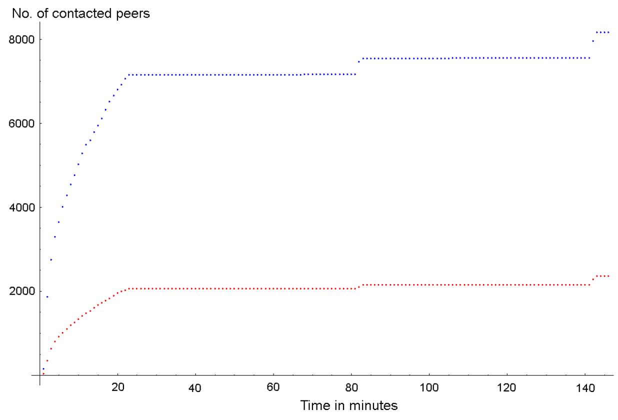


Figure 3: IP addresses contacted by a newly infected Storm node.

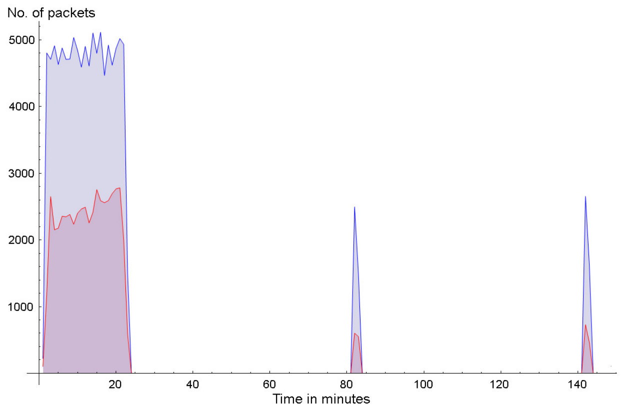


Figure 4: Network traffic pattern from initial infection onwards.

3. Search for an identifying hash (generated daily) to find command information or update location.
4. Parse and decrypt the search result sent from one of the nodes on the network. The output of this step is a URL.
5. Download and execute the file pointed to by the URL.

The network bootstrap process and subsequent status updates generate significant amounts of traffic on the network. Figure 3 shows the number of IP addresses contacted when sending (blue) and receiving (red) peer-to-peer information. Figure 4 shows the number of network packets being sent (blue) and received (red) by a system recently infected with a variant of the Storm worm. The first 20 minutes are spent connecting to the peer-to-peer network. Afterwards, an update of the available peers is performed once every hour. These updates can be seen in Figure 4 as peaks after 80 and 140 minutes of activity on the graph.

The Overnet protocol relies on 32-bit hashes [3] to identify nodes uniquely on the network. When a node searches for information on the network, it generates an MD4 hash using the name of the information and then searches for a match to that hash. To search for the hash, the node sends search queries to peers in its list of known peers that have a hash closest to the one it is looking for. These hashes are XORed with the one the node generates to find the ‘distance’ between them – the lower the value, the closer the hash.

STORM TRACKING

When we discovered that Storm was using the Overnet protocol, and a fixed set of hashes generated by a key within the binary, we were able to create a program that let us monitor Storm worm communications transmitted on the Overnet network. This was essentially a set of Ruby classes that implemented part of the Overnet protocol. This allowed us to connect to and search for Storm-related information on the Overnet network, and gave us the capability to parse packet captures of Overnet communications. Using this method we were able to monitor the Storm worm network for more than four weeks, effectively finding update locations and fetching new samples as they were released.

Unfortunately (probably because there were a number of security researchers doing similar things) the malware authors later modified their code and changed the communication scheme.

Around the month of May, the Storm worm started encoding its communication data inside strings that looked more legitimate for peer-to-peer file sharing. An infected system would still perform a search for an MD4 hash on the Overnet network but the answer would come as a set of strings similar to file names and sizes (e.g. ‘14173.mpg;size=83526;’). It seems that the update mechanism was changed and update locations were no longer transmitted over the peer-to-peer network.

STORMY WEATHER

The Storm authors use a variety of means to hide their creations and hinder the reverse engineering process. A simple, but very effective way of hiding the functionality of an executable is to fetch the code for a certain behaviour only when it is needed. In variants discovered at the beginning of the year, Storm worm programmers decided not to include any payload in their malware other than an update mechanism [4].

If they wanted to use the botnet to send spam, they would issue an update containing all the code to send that spam and the addresses to which they wanted it sent. In the same way, a DDoS component with hard-coded victim address would be distributed via the update mechanism present in the Storm worm. This functionality allows the worm to maintain a measure of stealth against detection, and to give away nothing to malware scanners because all ‘attacks’ are planned offline, and a customized ‘update’ delivered on each occasion.

Storm’s authors have always used their own custom packers, but around mid-August, they released yet another version with a new packing method. The latest version of their packer calls unusual functions such as DragQueryFile from SHELL32.dll and CreateMDIWindow from MDI.dll. The return values of these function calls are used by the packer algorithm to decode the original program. This technique seems to be used to fool emulators that haven’t implemented the complete set of API calls made available by *Windows*. Some emulators will simply return 0 if a call is made to an unimplemented function. In this case, the packer

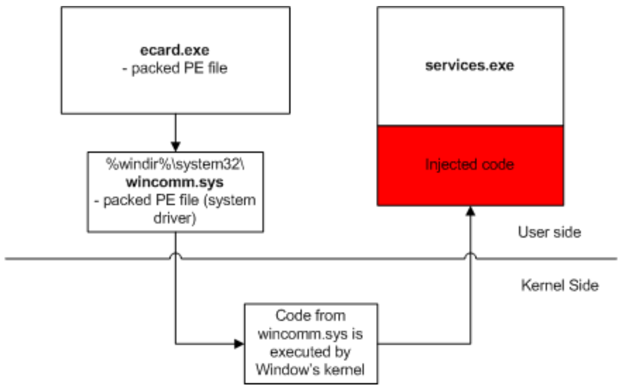


Figure 5: Example of Storm worm code injection.

will use a bad return value and the packed code will never be unveiled properly.

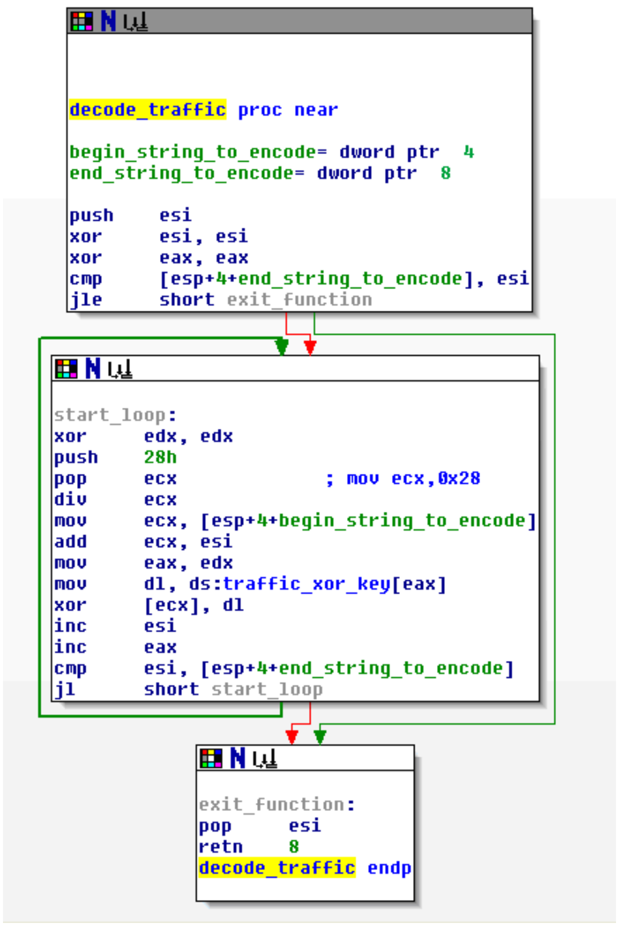
The following assembly code shows a technique used in the Storm worm packer to obfuscate the way it writes to memory. Instead of moving a modified value directly to the destination address with the mov operator, the packer changes the stack pointer and then pushes a 32-bit value on the stack. The real stack pointer (esp) is saved in the edx register. After that, the destination address is moved in esp. The value that needs to be written to memory is then pushed from eax to the target memory using the push operator. Finally, the good stack pointer value is restored in esp:

```
mov edx, esp
mov esp, edi
push eax
mov esp, edx
```

It is possible to hide the presence of programs and files on a *Windows* system by hooking specific APIs within the operating system and by injecting code in other processes. Figure 5 shows how a version of the Storm worm executes its code while keeping its presence hidden from the user and from some security solutions.

The malicious file enters the victim's system as a Portable Executable (PE) file. In most cases, the executable file will be packed, as previously described. When executed, it drops a *Windows* driver file in the %windir%\system32 folder. Before finishing its execution, a (malicious) driver is loaded by calling the CreateService and StartService system calls. This process is described further in [5]. The loaded driver will inject code from the kernel side into the services.exe process. The code injected in services.exe then connects to the peer-to-peer network in search of further payloads or software updates.

One of the more recent tactics has been to modify a *Windows* driver file to take advantage of the fact that it is loaded in memory every time the system boots. Storm modifies tcpip.sys, located in the *Windows* system folder, to load the malicious driver spooldr.sys. The malicious driver will, in turn, hide its presence through API hooking and inject code into services.exe.



```
decode_traffic proc near

begin_string_to_encode= dword ptr  4
end_string_to_encode= dword ptr  8

push    esi
xor     esi, esi
xor     eax, eax
cmp     [esp+4+end_string_to_encode], esi
jle     short exit_function
```

```
start_loop:
xor     edx, edx
push    28h
pop     ecx                 ; mov ecx,0x28
div     ecx
mov     ecx, [esp+4+begin_string_to_encode]
add     ecx, esi
mov     eax, edx
mov     dl, ds:traffic_xor_key[eax]
xor     [ecx], dl
inc     esi
inc     eax
cmp     esi, [esp+4+end_string_to_encode]
jl      short start_loop
```

```
exit_function:
pop     esi
retn    8
decode_traffic endp
```

Figure 6: XOR loop application.

In common with other modern malware, Storm tries to terminate a list of predefined processes belonging to anti-virus programs. If Storm is allowed to execute on a system, it will terminate those processes, and disable any anti-virus protection – so it's clear that proactive detection is critical.

```
0000    00 1c 42 00 00 02 00 1c    42 96 1d 6a 08 00 45 00    ..B..... B..j..E.
0010    00 35 00 23 00 00 80 11    57 99 0a d3 37 05 44 c4    .5.#.... W...7.D.
0020    5c 60 38 77 47 35 00 21    7d 72 10 a6 d8 9c 9b 2f    \'8wG5.! }r...../
0030    7d 45 b3 6e f9 61 62 2d    da c1 fc 72 63 3d e6 13    }E.n.ab- ...rc=..
0040    a8 54 46                                              .TF
```

Figure 7: Dump of a sent packet.

```
0000    00 1c 42 96 1d 6a 00 1c    42 00 00 02 08 00 45 00    ..B..j.. B.....E.
0010    00 1e 4f 63 00 00 ff 11    a3 8a 44 bc 42 4d 0a d3    ..Oc.... ..D.BM..
0020    37 05 7c af 38 77 00 0a    71 2b 10 a7                7.|.8w.. q+..
```

Figure 8: The answering packet.

RECENT EVOLUTION

Around the beginning of October 2007, the authors of the Storm worm made a significant modification to the communication protocol being used. They decided to encode the traffic by using a pre-shared key and XORing the traffic with that key. Figure 6 shows a disassembly of the XOR loop being applied to data before the network communication routine.

Figure 7 is a dump of a sent packet. Offset 0x2A is the protocol identifier (usually 0xe3 for Overnet) and the next byte should be the Overnet message type field (in this case, it should be 0x0c for Publicize).

The answering packet (showing that communications are encoded both ways) is shown in Figure 8. At offset 0x2A, we still have 0x10 as the protocol identifier and the next byte is 0xa7 instead of 0x0d for Publicize ACK.

This change in behaviour is quite significant since only systems that are infected with Storm will answer these encoded p2p messages – ironically making it easier to identify and disinfect infected hosts.

HARD RAIN GONNA' FALL

The developers behind the Storm worm seem to be members of an organized crime gang. In February 2007 they were observed using their network of infected hosts to conduct denial of service attacks against the infrastructure used by the competing Warezov botnet [6] – again used to send out spam.

However, the main functionality and final goal of the Storm worm seems to be to send pump-and-dump emails [7] in an attempt to accumulate large profits for its programmers and operators – although it's not clear if the operators are the same people profiting from the scams, or whether the operators are simply renting capacity to those scammers.

Pump-and-dump scams have become very popular in recent months, and according to some sources, the return on investment is high [8].

Fraudsters use the infected systems to send emails enticing recipients to buy typically very cheap stocks of a company, predicting a major upturn in the share price over the next couple of days. Before sending the spam, the scammers purchase large amounts of the company's stock and when the stock begins to gain value (as a result of the increased interest generated by their emails) they sell all their stock and cash out, often doubling their investment. After this large selling action, the stocks often crash and the other buyers (and even investors who didn't buy) lose what they invested [9].

LONG RANGE WEATHER FORECAST

Because the Storm worm has undergone such rapid, complex and challenging evolution, it has drawn a lot of attention from researchers investigating it. However, it seems that the developers behind the worm recently included an automatic denial of service capability, which will attack IP addresses that make too many queries to key systems on their infrastructure. It is therefore our advice to tread lightly!

There seems little doubt that the effort and development time put into malware like Storm is an ongoing trend, and will be the major challenge facing the anti-virus industry for the foreseeable future. Never in the field of malware has so much been released to so many by so few. (Our apologies to Sir Winston Churchill for the awful paraphrase.)

REFERENCES

[1] Wang, P.; Sparks, S.; Zou, C. An advanced hybrid peer-to-peer botnet. 2007. http://www.usenix.org/events/hotbots07/tech/full_papers/wang/wang.pdf.

[2] Data cross-validated with Storm worm chronology 26 September 2007. http://www.websense.com/securitylabs/blog/blog.php?BlogID=147.

[3] Maymounkov, P.; Mazières, D. Kademlia: a peer-to-peer information system based on the XOR metric. 2002. New York University. http://www.scs.stanford.edu/~dm/home/papers/maymounkov:kademlia.ps.gz.

[4] Trend Micro. War against NUWAR: fighting the latest profit-driven, multi-component, focused attack. December 2006. http://www.trendmicro.com/vinfo/secadvisories/default6.asp?VName=War+Against+NUWAR%3A+Fighting+the+Latest+Profit-driven%2C+Multi-component%2C+Focused+Attack.

[5] Hoglund, G.; Butler, J. Rootkits, subverting the Windows kernel. 2005. Addison-Wesley.

[6] Stewart, J. Storm worm DDoS. February 2007. http://www.secureworks.com/research/threats/storm-worm.

[7] Lemos, R. Imperfect Storm aids spammers. February 2007. SecurityFocus. http://www.securityfocus.com/news/11442.

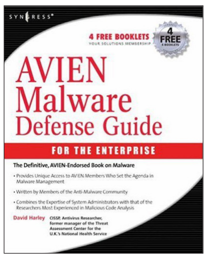
[8] Frieder, L.; Zittrain, J. Spam works: evidence from stock touts and corresponding market activity. March 2007. Berkman Center Research Publication No. 2006-11. http://ssrn.com/abstract=920553.

[9] Lee, A.; Harley, D. The Spam-ish Inquisition. September 2007. http://www.eset.com/download/whitepapers.php.

BOOK REVIEW

BIRDS OF A FEATHER...

Martin Overton
Independent researcher, UK



Title: AVIEN Malware Defense Guide for the Enterprise

Author: David Harley, *et al.*

Publisher: Syngress

ISBN 13: 978-1-59749-164-8

Pages: 540

Cover price: $59.95

The *AVIEN Malware Defense Guide* has been written by members of the AVIEN/AVIEWS online communities with the aim of passing on knowledge that they believe will be both interesting and useful to those involved in the real-world battle against malware in organizations.

The cover of the book claims that it will 'stop the stalkers on your desktop' and also provide:

- Complete coverage of the relationship between enterprise security professionals, customers, vendors and researchers.
- In-depth consideration of key areas of the 21st century threat landscape.
- System security and DIY defence using a range of specialist detection and forensic techniques and tools.

Meanwhile, the back cover states: 'AVIEN members represent the best-protected large organizations in the world, and millions of users. When they talk, security vendors listen: so should you.' So, after making such a bold statement, does the book deliver on the promises it makes?

UNDER THE COVERS

The book contains 11 main chapters and two appendices. It starts with a short biography of each of the contributors, before moving on to a very brief foreword, penned by Robert Vibert, who was administrator for AVIEN at the time the book was written.

This is followed by a preface and introduction, both written by the book's main author David Harley, in which an overview of each chapter is provided in a concise, but friendly manner.

CHAPTER AND VERSE

The start of each new chapter is marked by a cover page which presents a list detailing the content of the chapter. The publisher calls this list 'Solutions in this chapter' – even when the subject matter of the chapter relates to threats, rather than solutions. Every chapter concludes with a 'Summary', a 'Solutions Fast Track' and a 'Frequently Asked Questions' section.

Chapter 1, 'Customer power and AV wannabes?', is a nice gentle introduction, describing how AVIEN and AVIEWS started and how they have evolved over the years. It moves on to discuss how the anti-virus industry reacted, initially with some suspicion, to the formation of the groups, before realizing that AVIEN wasn't a vendor-bashing forum but a valuable information-sharing resource.

The reader is then asked 'So you want to be a bona fide computer anti-malware researcher?', which is followed by 'You should be certified' (not *that* sort of certified, even if eccentricity does seem to be a common trait in the industry), and the chapter finishes off by considering the question 'Should there be a vendor-independent malware specialist certification?'.

Chapter 2, 'Stalkers on your desktop', ups the pace a little by covering malware nomenclature – this is always a fun topic as vendors rarely agree on naming (in theory, yes, in practice, no). The chapter covers the CARO naming convention, as well as the Common Malware Enumeration effort led by *MITRE*. We then take a trip down memory lane with a look at the birth of malware. Viruses, trojans and worms are covered, as well as spam, rootkits and scams, and finally hoaxes and chain letters get the once over.

Chapter 3 is entitled 'A tangled web' and covers the threats that rely on HTTP, such as index hijacking and hacking into websites. It moves on to discuss browser vulnerabilities and attacks on DNS servers, such as DNS poisoning (pharming).

After looking at the threats the chapter turns its attention to some of the many solutions available as well as the testing of HTTP-scanning solutions, and covers 'malware and the web: what, where, and how to scan' and 'parsing and emulating HTML'. It also covers some of the legal issues associated with the business of blocking malicious threats from the Internet, looking specifically at patents and the all too common litigation between patent holders and anti-virus companies.

Chapter 4 is entitled 'Big bad botnets' and is essentially a compressed version of several chapters of another recent *Syngress* book on botnets (see *VB*, June 2007, p.7).

Chapter 5, 'Crème de la cybercrime', covers the changing face and motivation of malware authors, looking at both old-school virus writing as well as the more recent developments of the blackhat economy.

This is underlined by a couple of case studies which clearly show the current motivation of the bad guys involved in malware authoring and cybercrime. The chapter finishes off with a look into a virtual crystal ball and discusses what won't change, as well as the things that are more likely to happen. These include not only techniques such as social engineering, but also technologies, such as VoIP, credit cards and podcasts.

Chapter 6 is entitled 'Defense-in-depth' and describes the technique very thoroughly, explaining what it is and how to implement it. More importantly it also covers some of the areas that often get forgotten and even goes as far as covering malware laboratory procedures. For most corporate security personnel, this chapter will be indispensable. It is a real goldmine of useful material and very well thought out.

Chapter 7, entitled 'Perilous outsorcery', covers the thorny and emotive subject of outsourcing anti-virus services. It not only covers how outsourcing can best be achieved, but also some common mistakes and the importance of two-way communication.

The key thing that is made absolutely crystal clear in this chapter is that outsourcing anti-virus services is not a quick fix, even if your existing in-house anti-virus service isn't broken or badly sprained before you decide to pass this 'hot-potato' to your chosen outsourcer. I couldn't agree more.

Chapter 8 is entitled 'Education in education' and, not surprisingly, covers the subject of user education. However, it discusses it in a new and refreshing way, looking not only at the subject from an educationalist's perspective, but also the issue of security in education.

The chapter concludes with 'Not exactly a case study: the Julie Amero affair'– discussion of a contentious court case in the USA that has been hot news in the anti-malware industry since the story originally broke in January this year. The whole chapter is very well written and extremely well thought out.

Chapter 9, 'DIY malware analysis', is the most technical chapter in the book as it deals not only with analysing malware using web-based tools, but also using debuggers and disassemblers for static code analysis. It also covers, in some depth, dynamic analysis in virtual environments such as *VMWare* and *VirtualPC*, and the use of behavioural monitoring tools. Packers, memory dumping and forensics all get some coverage too.

This chapter is definitely not for the faint-hearted or those that have never handled live malware before. Geeks/nerds will love it.

Chapter 10 is entitled 'Antimalware evaluation and testing' and details how anti-malware tools should be tested, and by whom (for example by *Virus Bulletin*). It also covers how *not* to test, giving examples of how poor some of the non-specialist testing can be. However, it does also offer an evaluation checklist which could be used as a handy guide for in-house product evaluation, if you really must do it yourself.

The chapter also discusses the importance of researcher ethics and sample verification, and is rounded off with a look at independent testing and certification bodies, including the likes of *Virus Bulletin*, *ICSA Labs* and *AV-Test.org*, as well as several others.

To demonstrate that there is such a thing as a perfect anti-malware solution the chapter summary includes a link to Dr Alan Solomon's 'Perfect AV' article (http://members.aol.com/drasolly/perfect.htm).

Chapter 11 completes the book with a look at 'AVIEN and AVIEWS: the future'. This chapter is really a quick summary of many of the things covered in the book as well as a look at how AVIEN and AVIEWS have adapted to the changes in the threatscape since their inception, and how they continue to adapt.

Not surprisingly the chapter ends with the suggestion that if the reader isn't already a member of AVIEN/AVIEWS they consider joining. If this suggestion were from a commercial company I would be inclined to call it shameless marketing. However, as AVIEN/AVIEWS is a non-profit organization, and I have found the benefits it provides to be very valuable, I say come and join in.

CONCLUSIONS

So, what impressions am I left with after reading the first AVIEN/AVIEWS book? My overriding impression is that this book is very well written; the whole book comes together and flows very well – which can be a difficult feat when a book has several different contributors.

The book eases the reader in gently, starting with non-technical chapters and building to some very technical ones towards the end of the book.

The pedigree and diversity of the contributors involved in this book makes it a very readable, informative, and accurate reference guide for all interested parties, be they new to the fight or old hands.

The book delivers on many of the promises it made. In fact, I would say that this is the best general malware/anti-malware book currently available, and it should be a mandatory read for anyone new to computer security in general, and anti-malware specifically.

I'm already looking forward to the second (updated) edition.

PRODUCT REVIEW

ESET SMART SECURITY

John Hawes

ESET has for some time been the small company with the big voice, making its presence felt in the early days through excellent test performances (both in terms of detection levels and speed), and more recently through a series of aggressive and stylish marketing campaigns. As a result, the company's anti-malware product *NOD32* has built up a strong and devoted following, particularly among more technically minded users for whom the idiosyncrasies of a rather unusual interface are more than outweighed by the product's impressive power and low overheads.

Now, however, the firm is setting its sights firmly on the broader market. *ESET*'s latest offering, *Smart Security* (*ESS*), is a multifunction home-user product which promises to be formidable competition for the range of Internet security suites already on the market.

ESET was founded in 1992 in Bratislava, Slovakia, and now has offices in the US, UK, Argentina and the Czech Republic. The company's flagship product *NOD32* (these days officially known as *NOD32 Antivirus System*) has for some time been the top-performing product in *Virus Bulletin*'s comparative reviews – having amassed more VB100 awards than any other product.

Rarely missing a sample in any of the *VB* test sets, *NOD32* also trounces its rivals regularly in speed tests, often leading the field by some considerable margin – and while recently some of those rivals have started catching it up, *NOD32* remains among the very fastest month after month. These test results are confirmed by other independent bodies, with *AV-Comparatives* granting the product its 'Best of 2006' award.

So, with such a strong history and reputation, *ESET* presents an impressive challenge to its bigger rivals – one that, for many market watchers, has been held back only by the perceived 'techiness' of the product design. Though always incorporating some flair and panache, there has long been a hint of obscurity in *NOD32*'s GUIs, exemplified by the lingering use of acronyms for the various sections of the product – AMON for the on-access file monitor, IMON for the Internet guard etc.

With a new product aimed squarely at the broader, less technically minded market, and backed up by a new version of the detection engine, many have been eager to find out whether *ESET* has managed to shake off the techiness and produce a truly accessible interface to match the wonders of the technology beneath. I counted myself lucky to be among the first to find out.

WEB PRESENCE, INFORMATION AND SUPPORT

ESET's main website, www.eset.com, is an attractive place dominated by the company's dark green colour scheme. The slightly spooky, sci-fi eye symbol of *NOD32* features prominently, along with links to a 30-day free trial. The front page also carries details of the numerous accolades, awards and compliments received by the firm and its products, with the news section carrying details of a recent honour bestowed on the company's San Diego-based US office for its recent rapid growth.

A graph of recent infections shows the malware trends seen over the last few hours, with links provided to the company's *VirusRadar* statistics site, free removal tools and online scanner system. Like so many similar products, the online scanner system uses ActiveX and thus requires *Internet Explorer* (5 or later), along with all the associated risks of those technologies, but it does provide malware removal and detection of 'potentially unwanted' nasties.

Elsewhere on the site are the usual swathes of information on the range of products available, which cover a wide set of platforms including *Microsoft* offerings as wide apart as the aged DOS and the very latest *Vista*. File and email server editions are also offered, as well as centralized management and reporting tools in the enterprise versions.

Perhaps most interesting in this section are some pages comparing *NOD32* with some of its rivals. A table shows *NOD32*'s considerable lead in the number of VB100 awards it has received, its scanning speeds as recorded in *Virus Bulletin* tests, as well as its unmatched record of not having missed any WildList samples in a *VB* comparative review since May 1998. Snippets from *AV-Test* and *AV-Comparatives* show a similarly impressive record of heuristic detection – something else for which the product is justifiably renowned – with both testing bodies rating *NOD32* best in the field at spotting unknown threats with old signature databases.

Alongside areas providing trial downloads and online purchasing, the 'Threat Center' brings together various forms of information on the malware problem, with articles and white papers backing up some simple descriptions of malware types and the obligatory malware encyclopaedia. While not up to the standard of some of the bigger encyclopaedias provided by larger companies, and somewhat hampered by an inelegant layout, the encyclopaedia provides some useful information on a number of leading threats for those in need. The main page also carries links to the online scanner, a range of useful external malware resource sites and the *VirusRadar* site, www.virusradar.com, which features up-to-the-minute statistics on prevalence, new threats and outbreaks.

The company's blog, also found here, differs from many vendor blogs in that it does not tend to focus on the latest discoveries, but instead provides a stream of insightful pieces, providing anecdotes, tips and advice on staying safe from malware and scams in the online world.

The support section provides a well-stocked FAQ and links to manuals and so on, as well as a support request form which promises free 24-hour response. A similar form is built into the product, enabling detailed system information to be sent along with the request, which can be configured and checked before sending. The new *Smart Security* product, due for release at around the same time as this review, is yet to be mentioned in any of these provisions. Experienced *NOD32* users will know, however, that the true home of *ESET*'s support is the official forum hosted by *Wilder's Security*, a busy place where queries of all types are answered rapidly by both fans and trained company representatives. This includes a section on the beta versions of the new product, which will presumably soon morph into coverage of the full release version.

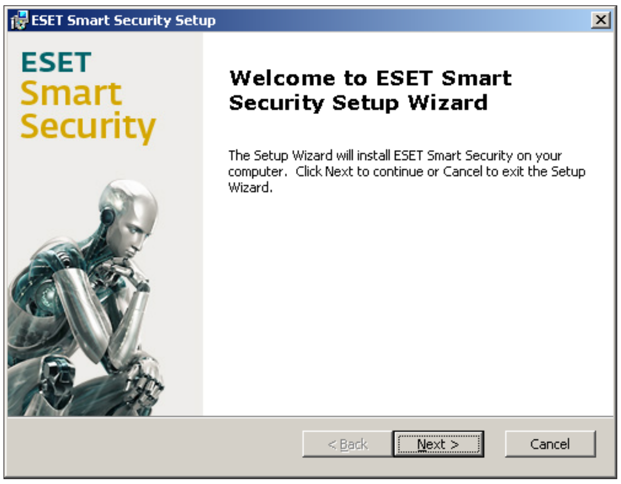
INSTALLATION AND SETUP

As this product is so new, I was not able to see a full boxed version to check out the packaging, CDs etc., although I was given a preview of some very attractive cover artwork for the boxes. Instead, the product was provided in the form of two msi installers, for 32- and 64-bit systems; the packaging pictures revealed that there are to be separate home and business editions, for both *ESS* and the less complete but still potent package of *NOD32*.

The initial splash screens of the installer set the tone of the new product. The eye symbol used by earlier versions of *NOD32* has been replaced by a very glossy and attractive symbol of the new world – a shiny silver android, shown in a number of pensive poses, representing the 'intelligence' of the scanning system, analysing files for unknown threats using those highly regarded heuristics.

Beneath this rather beautiful skin, little seems changed in the installation process, with the familiar questions about settings, updating and enabling various optional components.

These include the ThreatSense.net system, which submits detections back to base to provide information on outbreaks as well as allowing new heuristic detections to be analysed more thoroughly, with full detection for new items then pushed back out to fellow users. There is also the option to enable or disable detection of 'potentially unwanted' files and products, neither of which is checked by default. Both of these settings can, of course, be changed later through the normal interface. The most taxing part of my first few



Welcome to ESET Smart Security Setup Wizard

The Setup Wizard will install ESET Smart Security on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

installations was the entry of user credentials to activate the product's licensing, but even these can be put off until later if desired, allowing for a trial install.

The setup process is fast and smooth, and the various requests for information put clearly and simply with little to worry about for the novice user. A more advanced setup process is also available for those who wish to fine-tune the likes of proxy settings, firewall configuration etc. from the off. In practically no time the thing was fully installed and ready to go, with no reboot requested, although in one instance, running 64-bit *Windows Vista*, this did seem to be necessary to activate the on-access protection fully (an issue that I understand has been fixed in the final release build).

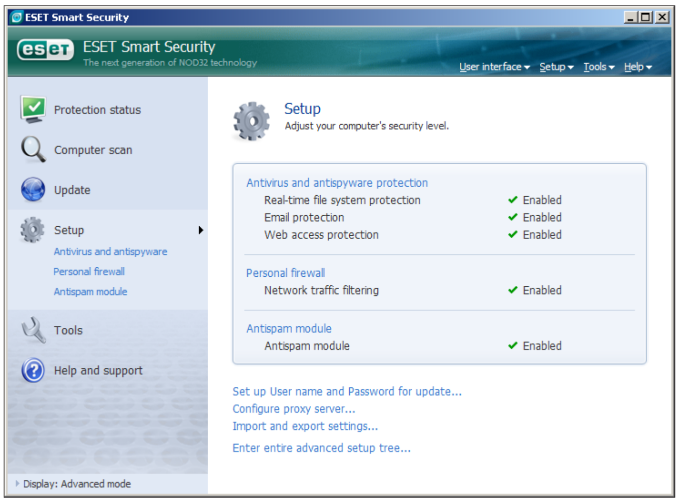
OPERATION AND DOCUMENTATION

With the main interface up and running, the change from the previous product is quite startling. Gone are the rather funky separable windows with their cryptic menu entries. Here instead is a much more conventional window, reflecting the general trend of such things and likely to be familiar in layout to anyone who has used a modern Internet security suite. Down the left-hand side are the major categories of information, starting with an overview of general protection status, indicating which modules are running at full power and accompanied by some simple figures on update versions, licensing and blocked attacks. Next is a scan section, where on-demand checks of the local system or individual areas can be run; an update area with more details on the latest detection data; and a setup page which allows quick changes to the operation of the product, such as password-protecting the controls, and switching on various parts of the functionality, including the on-access scanner and firewall. All of these are nice and simple, with no challenging technical jargon or overwhelming reams of numbers, and the layout is logical and easy to navigate. The functions offered are well thought out, catering for most of the everyday needs of the home user.

The interface can be flipped into a more advanced version, again using a simple and obvious button. This provides an extended version of the same set of pages, now subdivided where appropriate to access more detailed information and further control of the anti-malware, firewall and anti-spam subsections. An additional tab entitled 'tools' allows things like logging, quarantined files and scheduled jobs to be managed. Many of the control buttons in this area open up a new window containing the holy grail of configuration: the full setup pages for every conceivable area of the product, allowing the most dedicated fine-tuner to fiddle away to his or her heart's delight.

There is configuration available here for all the standard aspects of the product. Scanning for viruses, spyware, trojans, etc. targets email and web vectors as well as standard files on disk and in memory (functionality that has been part of the *NOD32* offering for some time), with *Outlook* monitoring set by default and other POP3 clients also supported. In the web filter, browsers and http ports can also be added and tweaked as needed, and software seen to be rendering clickable links (such as *Microsoft Word*) will automatically be counted as a browser and have its associated traffic watched over.

The whole thing seems pretty rational and nicely laid out, causing me only a few moments of confusion. The 'cleaning' mode of various modules is configurable only via a simple three-step slider, which offers a choice of no cleaning at all, 'strict' cleaning at the other end of the scale, and the default in the middle. As far as I could tell, the 'strict' mode deletes whole archives if an infected file is found inside, while the default does not, and the two are otherwise indistinguishable. However, the popups alerting the user to infected files offer more fine-tuning of actions,



Setup
Adjust your computer's security level.

Antivirus and antispyware protection
Real-time file system protection ✓ Enabled
Email protection ✓ Enabled
Web access protection ✓ Enabled

Personal firewall
Network traffic filtering ✓ Enabled

Antispam module
Antispam module ✓ Enabled

Set up User name and Password for update...
Configure proxy server...
Import and export settings...
Enter entire advanced setup tree...

which can then be applied more permanently. The only other issue I had with the configuration pages was the apparent absence of tweaking for the right-click scanning mode, which was fairly prominent in earlier *NOD32* versions (it could well have been in there somewhere, but was not immediately obvious). Of course, this sort of thing is of more interest to testers than to the average desktop user.

Help for this type of issue is provided of course, with the focus still firmly on the less experienced user. Everything is clear and simple, in plain language, and coverage is remarkably thorough. Links from within the product to the appropriate pages of the help are sparse, but seem present where most needed.

MALWARE SCANNING AND DETECTION

Scanning with *NOD32* has long been one of the least arduous tasks required of a VB100 tester. With misses vanishingly rare, crashes and slowdowns similarly unlikely, and configuration systems invariably well-tailored to the tester's needs, the product has long been something of a dream to work with. I must say that the new version took a little getting used to before its full potential could be unleashed. The simplified interface, even when toggled into the 'advanced' mode, offers little by way of configuration. Full system scans can be carried out fairly easily, and the right-click option allows for easy checking of individual files and folders, but for my somewhat specialist needs much plunging into the full setup page was required to ensure things were scanned as I wanted them to be. With the layout of this area easily mastered, however, this was no real chore.

A few scans of the *VB* test sets showed excellent detection ability as anticipated. The other striking aspect of *ESET*'s products, the superb scanning speeds, seemed if anything to have improved – scans blasted through in barely any time at all. Doing some further investigation into this phenomenal pace, I ran the product over the speed tests included in recent VB100 comparative reviews, comparing it with a version of *NOD32* from a few months ago – of course, this is hardly a fair test, as those few months would have added much by way of detection data, and the new product has other things to do besides scanning. Remarkably, scans across most of the sets showed almost identical times despite the larger detection coverage of the new version, with some sets and platforms actually showing small improvements, like a sprinter shaving a few milliseconds off their personal best. The executable set and *Windows Vista* showed the most improvement, implying that *ESET* has been fine-tuning its scanning engine to focus on the most commonly infected items and what is almost certain (despite calls for boycotts in some areas) to be the platform of the future.

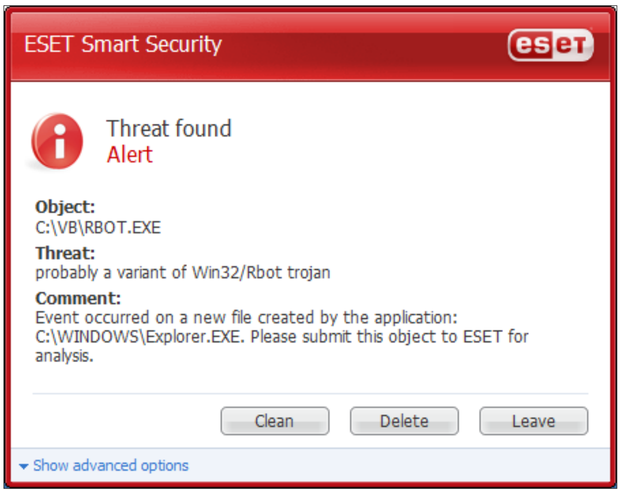
The heuristic capabilities of *ESET*'s product have long been highly regarded. They came through strongly again here, with large numbers of samples detected generically or as as-yet-unknown malware variants. The pensive android figure which adorns the product is intended to signify its built-in intelligence, picking apart files looking for suspect instructions, and at times, despite the speed, this could be seen in action. On a tiny handful of samples, including some in the latest WildList (released some time after the product was sent in for review), the infected files were missed, but on some of them the scan seemed to pause, looking extra hard at the file as if it had spotted something a little off but couldn't quite put its finger on it. It seemed likely that if the heuristic paranoia level could be turned up a notch or two these files would have been labelled as suspect. Once passed by the scanner, some naughty behaviour, including placing files in system folders and initiating network connections, was then allowed – indicating that perhaps even with the product's excellent detection rate a full-on intrusion prevention system, or more secure firewall settings, may be useful to fend off the very latest and most cunningly emulation-proofed threats. Retrying with a more up-to-date version showed all the missed samples were identified accurately and with ease.

The 'ThreatSense.Net' system tied in with the heuristics is designed to send samples of suspect files back to base for further analysis, providing group immunity and early warning of outbreaks. This is a technique which *ESET* adopted early on, and one which is becoming an increasingly popular and useful part of anti-malware products, as it allows for fast response to the ever-accelerating emergence of new threats. The behaviour of *ESET*'s system, including the quantity and type of data sent, is highly configurable and defaults to sending files during updating periods only, and checking for confirmation from the user first, but it can be set to send automatically as soon as a suspicious file is spotted.

OTHER FUNCTIONALITY

Smart Security shies away from the 'suite' title so popular with multifunction products these days, instead preferring to be considered a single integrated whole offering several security services from a single point. Updates reflect this lack of modularization, making no differentiation between definition updates and improvements to the detection engine. The extra components, not featured in previous versions of *NOD32* or the new one that is sister-product to this, are essentially the client firewall and spam filter.

The firewall is the biggest addition to the new product, having become pretty much a necessary part of any security product these days and something that is generally



Object:
C:\VB\RBOT.EXE
Threat:
probably a variant of Win32/Rbot trojan
Comment:
Event occurred on a new file created by the application: C:\WINDOWS\Explorer.EXE. Please submit this object to ESET for analysis.

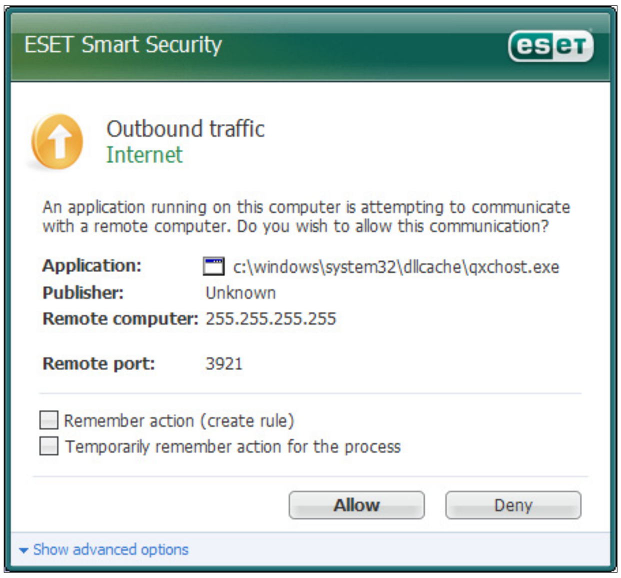
acknowledged as essential to safe networked computing. The default setting, chosen during installation, is 'automatic' mode, implying that the system decides what is safe and what is not. This approach has been tried by many products, generally needing at least a few screens of setup queries, and can be fairly successful, although there will always be some piece of software the developers missed out when setting up the default rules. The more standard tactic – prodding the user with popups every time a new item needs to connect to the network – is a more cautious approach, and there can be no denying that this method often only serves to frustrate people who don't have the experience to understand whether a program can be trusted just from its filename.

The settings of the *ESS* firewall seemed admirably well-tuned, with 'standard' outgoing connections allowed after a little filtering and automatic blocking of inbound connections that have been initiated externally. The only question I received from the firewall, once I had allowed it to go along with the default automatic setting, was whether I wanted to allow filesharing within the local network, and even this was put in plain and simple language. Of course, more paranoid users, or those with less 'standard' requirements will want to investigate the deeper configuration, which is also possible.

The firewall page of the main interface provides some handy buttons to shut down all network activity in an instant, or to disable the firewall completely, and full configuration is accessed via the advanced setup screen. Two further modes are available: an 'interactive' setting where users are prompted to allow or deny connections as they occur, with the usual option to create a permanent rule or simply allow a one-off connection, and a 'rule-based' mode where full sets of rules are defined in advance and applied to the letter, with anything not specifically allowed by the rules barred access to the network. Obviously, both require some technical knowledge of what is needed and what can be done without, but if configured properly should provide an even higher level of security. The firewall integrates with the web side of the malware scanner, so rather than blindly obeying the rules laid down traffic is monitored for malicious code passing through via normally permitted channels.

The anti-spam side of things is again pretty straightforward. A single page in the advanced setup screen allows some minor tweaking of the behaviour – actions on spotting spam, automatic whitelisting of addresses from local address books or from outgoing messages and so on, while a statistics page in the main GUI shows the number of emails processed and what was done with them. The system currently only supports *Microsoft* email clients *Outlook*, *Outlook Express* and *Windows Mail*, and has a standard training system to allow the user's emailing behaviour to influence the basic filtering rules.

There are no additional bells and whistles here – not for *ESET* the hurried system clean-up tools, the encryption and data-leak prevention, or the tricky parental controls that are being bundled into various competing products. These things are, perhaps wisely, left to specialists in those fields. *ESET* concentrates instead on providing what it knows best: top-quality malware detection and removal, protection from the ever-growing deluge of nasties bombarding users via the web, email and unprotected networks. Items listed as modular extras in many other products, such as rootkit detection and advanced heuristics, are here quietly rolled in with the body of the product, as a necessary part of what it does rather than a tacked-on adjunct. The absence of a full-blown behavioural analysis or intrusion-prevention system is perhaps the only thing that seems significantly missing, and this seems likely to make its way into a future version of the product.



Outbound traffic
Internet

An application running on this computer is attempting to communicate with a remote computer. Do you wish to allow this communication?

Application: c:\windows\system32\dllcache\qxchost.exe
Publisher: Unknown
Remote computer: 255.255.255.255
Remote port: 3921

☐ Remember action (create rule)
☐ Temporarily remember action for the process

CONCLUSIONS

ESET seems to have upped the ante considerably here. While *NOD32* has always been a cutting-edge product in technical terms, usability has seemed very much an afterthought in previous versions, at least as far as general users were concerned. In a sudden leap, this new product has shaken off the complexity and actually taken something of a lead in terms of ease of use. The default settings put in place in a few easy steps during the installation seem ideal for general purpose use, information and alerting is clear and helpful without ever seeming intrusive or scare-mongering, and detailed configuration is about as straightforward and painless as such things can be.

With the user experience reduced to an installation process, and possibly the occasional alert as another attack is spotted and blocked effortlessly, the unobtrusiveness is further enhanced by the product's legendary lightning speed, miraculously unimpeded by the additional protection, keeping overheads to a level barely noticeable on modern high-powered systems. Lacking full behavioural monitoring may help minimize the slowdown, but with the level of in-depth analysis and emulation going on during scanning the performance is most impressive.

As vendors release their latest 2008 product ranges, the addition of new supplementary modules seems to be all the rage, most suites now sprawling with diverse functionality, often at the expense of user-friendliness and occasionally posing dangers of their own. *ESET* has resisted the temptation to sprinkle in too many extras, focusing instead on the core requirements of a security system. Covering all the essential bases with a smoothly integrated set of protective barriers, the combination of top-of-the-range detection, response time, heuristics and throughput with excellent presentation and design will make *Smart Security* pretty hard to beat.

Technical details

ESET Smart Security was variously tested on:

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

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

AMD Athlon64 3800+ dual core, 1 GB RAM, running *Microsoft Windows XP Professional SP2* (32-bit) and *Windows Vista* (64-bit).

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

END NOTES & NEWS

The CSI 34th Annual Computer Security Conference will be held 5–7 November 2007 in Washington, DC, USA. The programme and online registration are available at http://www.csi34th.com/.

E-Security 2007 Expo & Forum will be held 20–22 November 2007 in Kuala Lumpur, Malaysia. For event details and registration see http://www.esecurity2007.com/.

The Chief Security Officer (CSO) Summit 2007 will take place 28–30 November 2007 in Amsterdam, the Netherlands. The summit, entitled 'Security strategy to steer your business', offers participants the opportunity to tackle fraud management challenges in a hassle-free environment, surrounded by colleagues. A speaker panel will share direct experiences, successes, and tips gained from managing successful security projects. For details see http://www.mistieurope.com/.

AVAR 2007 will take place 29–30 November 2007 in Seoul, Korea. This year's conference marks the 10th anniversary of the Association of Anti Virus Asia Researchers (AVAR). For the full agenda, online registration and hotel booking see http://www.aavar.org/avar2007/.

SecureGOV 2007 takes place 2–4 December 2007. The fifth annual SecureGOV 2007 strategic intelligence council meeting will offer senior government IT, security, privacy and defence officers an insight into the latest developments critical to maximizing the protection of information resources, networks and critical infrastructure. For details see http://securegov.info/.

The 23rd ACSAC (Applied Computer Security Associates' Annual Computer Security Conference) will be held 10–14 December 2007 in Miami Beach, FL, USA. 42 refereed papers, six case studies, three panel sessions and a 'work in progress session' will cover a range of research topics, from security for P2P and mobile computing to malware and forensics. For details see http://www.acsac.org/.

Black Hat DC 2008 Briefings and Training will be held 11–14 February 2008 in Washington, DC, USA. The conference will focus on wireless security and offensive attacks in addition to the core set of training sessions. A call for papers for the Briefings closes 4 January 2008. For full details and registration see http://www.blackhat.com/.

The SecureLondon Conference on emerging threats will be held 4 March 2008 in London, UK. Attendees will be given an overview of the interaction between web, spam and malware, with a focus on specific campaigns. Sessions will engage in the devastating effects and developments of DDoS attacks and how to avoid them, email encryption and the social engineering threat communities pose to a company. For further information see https://www.isc2.org/cgi-bin/events/information.cgi?event=48.

Black Hat Europe 2008 takes place 25–28 March 2008 in Amsterdam, the Netherlands. Registration opens 1 November, and a call for papers closes 1 February. For details see http://www.blackhat.com/.

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

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

Black Hat USA 2008 takes place 2–7 August 2008 in Las Vegas, NV, USA. Online registration and a call for papers open 1 January 2008. For details see http://www.blackhat.com/.

VB2008 will take place 1–3 October 2008 in Ottawa, Canada. A call for papers will be issued shortly, details of which will be available at http://www.virusbtn.com/. Enquiries relating to any form of participation in the conference should be directed to vb2008@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)1235 531889
Email: editorial@virusbtn.com Web: http://www.virusbtn.com/

vb Spam supplement

CONTENTS

NEWS & EVENTS

SUPERMARKET SWEEP

Court documents have revealed that phishers nearly managed a whopping $10 million supermarket (bank account) sweep earlier this year.

Supervalu Inc., one of the largest grocery retail chains in the US, fell for a simple scam in which scammers sent the company emails purporting to be from two of the chain's approved suppliers, *American Greetings Corp.* and *Frito-Lay Inc.* In each case the email stated that the supplier had changed its bank account and requested that the retailer update its details and make all future payments into the new account. Over the next four to five days *Supervalu* made several deposits into the fraudulent accounts, with over $6.5m being transferred into the fake account associated with the *American Greetings* scam and $3.6m transferred into the account associated with the *Frito-Lay* scam.

Fortunately, while *Supervalu* staff were not on the ball enough to question the authenticity of the two emails, they did eventually work out that the bank accounts into which they had deposited more than $10m were bogus. *Supervalu* notified the authorities and the accounts were frozen before the scammers could withdraw the funds.

A *Supervalu* spokeswoman attributed the quick discovery of the fraud to the company's internal controls and processes – perhaps the company will now look at including better security training for its employees in those internal controls and processes.

MELISSA HAS USERS CAPTCHA'D

Spammers have spotted a new opportunity for getting humans to help them get past the CAPTCHA tests put in place to prevent illegal use of webmail accounts: promise users a series of photographs of an increasingly scantily clad woman for every CAPTCHA they complete.

The user is offered the opportunity to see virtual stripper 'Melissa' wearing progressively fewer clothes in a series of photographs – provided they correctly complete an accompanying CAPTCHA.

According to researchers at *Trend Micro* the CAPTCHAs are taken from *Yahoo Mail*'s signup screens – in completing the CAPTCHAs the recipients of the spam are inadvertently providing the spammers with webmail accounts that can be used to send more spam or for other nefarious purposes.

The striptease is part of a trojan variously named CAPTCHA.a, Captchar.a, and RompeCaptchas.A and is thought to be either part of a multi-stage attack, or encountered as a drive-by web exploit.

THE SOUND OF SPAM

Last month saw pump-and-dump spammers try out yet another file type for getting their message across to the gullible: MP3 audio files.

As the success of pump-and-dump scams relies on victims investing in shares, getting the name of a particular company lodged in recipients' minds is the main aim of the spam. Having already tried a variety of ways in which to display their messages, the latest trick is to send out MP3 audio files in the guise of tracks by popular musicians – the files in fact contain a voice, described variously as 'monotone' and 'rusty sounding', which advises listeners to invest in a particular company. *MessageLabs* reports having seen 15 million MP3 spams during October.

EVENTS

TREC 2007 takes place 6–9 November 2007 at NIST, MD, USA. For details see http://plg.uwaterloo.ca/~gvcormac/spam.

Inbox/Outbox takes place 27–28 November 2007 in London, UK. See http://www.inbox-outbox.com/.

The MAAWG 12th general meeting, open to members and non-members, will be held 18–20 February 2008 in San Francisco, CA, USA. See http://www.maawg.org/.

The 2008 Spam Conference takes place 27–28 March 2008 in Cambridge, MA, USA. Proposals for papers, tutorials or workshops will be accepted until 1 March 2008. See http://spamconference.org/.

FEATURE

EVADING SPAMASSASSIN WITH OBFUSCATED TEXT IMAGES

Battista Biggio, Giorgio Fumera, Ignazio Pillai, Fabio Roli and Riccardo Satta
University of Cagliari, Italy

Most spam filters consist of a set of modules which analyse different characteristics of an email (sender's address, header, body, attachments) to determine whether to label it as spam or legitimate mail (ham). In many filters the module devoted to the analysis of the email's textual content is based on statistical text categorization techniques. The application of such techniques to the spam-filtering task has been widely investigated by the machine-learning community over the past ten years (see for instance [1–4]).

To circumvent filtering modules based on text analysis, spammers started to embed their messages into attached images – this trick is called image-based spam (or image spam). Moreover, text images are often obfuscated using different techniques to render OCR tools ineffective without compromising human readability.

To deal with image spam, modules based either on OCR tools or on low-level image processing techniques have been introduced in spam filters. However, there is not yet a clear understanding of the effectiveness of image spam with obfuscated text in evading anti-spam filters that are based on OCR tools. As a consequence, there is also a lack of clear guidelines for the development of filtering modules against image spam.

In this work we focus on *SpamAssassin* [5], one of the most well known and widespread open-source spam filters, and provide a thorough and systematic analysis of its vulnerability to image spam with obfuscated text when an OCR-based filtering module is used. To this aim, we used a dataset of real spam emails with artificially generated images. The images were obtained by reproducing three kinds of the most commonly used text obfuscation techniques observed in real spam emails, and by varying the degree of obfuscation of each image in a suitable range.

We assessed both the performance of the whole *SpamAssassin* filter and that of the stand-alone OCR-based module. Two open source OCR tools used in *SpamAssassin* were considered: gocr [6] and tesseract [7]. Finally, we evaluated the *SpamAssassin* performance on a dataset of real spam emails with real attached images. The results of our analysis provide some useful insights into the effectiveness of OCR-based modules in spam filters, as well as some suggestions for the development of effective image spam-filtering modules.

1 SPAMASSASSIN ARCHITECTURE

SpamAssassin is made up basically of a set of 'if-then' rules, each one of which is dedicated to a different characteristic of an email which can be useful to determine either its spamminess or its hamminess. A score is associated with each rule: higher scores denote a higher degree of spamminess. The scores of the rules which fire (rules whose antecedent is true) are summed up, and if the sum exceeds a predefined threshold (whose default value is 5) the email is labelled as spam (see Figure 1). The score of each rule belongs to a given range (possibly different for each rule).

To improve the effectiveness of the filter on their specific email traffic, users can change the score range either manually or by using a built-in procedure named mass-check [8], which must be carried out on a user-defined dataset. The mass-check procedure sets the score range by taking into account the detection rate and the reliability of each rule. It is worth pointing out that the scores of *SpamAssassin* rules can either be binary or continuous-valued.

The *SpamAssassin* architecture is easy for users to understand and is extremely flexible, since it is possible to plug and unplug rules without changing the global configuration of the filter, i.e. without changing the score of the rules (although it should be pointed out that leaving weights unchanged as the rule set is modified may not be the best choice).

Among the set of rules available in *SpamAssassin*, there are currently four plug-ins dedicated to image spam: OCR plug-in, OCRtext, FuzzyOCR and BayesOCR.

The OCR plug-in looks for a set of predefined keywords in the text extracted by an OCR tool (either gocr or tesseract can be used) from the images attached to an email. A binary scoring system is used: if at least one keyword is found, the plug-in sets its score to the value of 3, otherwise its score is set to 0 (these are the predefined values). OCRtext uses the same idea as the OCR plug-in, but also checks some other properties of the attached image unrelated to content (such as aspect ratio and file size). FuzzyOCR works similarly to OCRtext, but carries out a fuzzy matching between

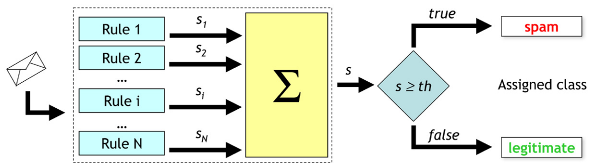


Figure 1: SpamAssassin architecture. The scores associated with the firing rules are summed up and if the sum exceeds a predefined threshold the email is labelled as spam, otherwise it is labelled as ham.

keywords and the extracted text, which in principle makes it more robust to OCR errors. Its scoring system is different as well. BayesOCR sends the extracted text to the text classification module of *SpamAssassin*, using the corresponding score. It is based on the work described in [9].

In our experiments we chose to use the OCR plug-in since it does not perform any pre-processing on the attached image, and carries out a simple keyword matching between the extracted text and a list of keywords in its database. This allowed us to assess directly how obfuscation techniques affect the performance of an OCR-based filtering module, and consequently the overall performance of the spam filter, without the need for taking into account the influence of other techniques (like image pre-processing or text categorization).

Finally, it should be pointed out that *SpamAssassin* has four different working configurations obtained by using or not using the text classifier based on the naïve Bayes classification technique (which is often used in spam filters, see for instance [1, 3]), and by using or not using Internet-related rules, e.g. DNS blacklist checking. In our experiments we assessed the performance of all four working configurations, using the default scoring system for each rule.

2 AUTOMATIC GENERATION OF SPAM IMAGES WITH OBFUSCATED TEXT

To carry out a systematic analysis of the effectiveness of image spam with obfuscated text in evading detection by a spam filter with an OCR-based tool, it is necessary to collect a large dataset of spam emails with attached images including representative kinds of obfuscation techniques.

Unfortunately, however, no freely available benchmark dataset with these characteristics exists. In particular, freely available datasets of spam emails were collected when image spam was not yet widespread and thus contain at most a negligible number of messages with attached images. Moreover, even in a personal archive of spam emails it is difficult to find a representative set of obfuscation levels. For this reason we developed a software module for generating artificial spam images characterized by different kinds of obfuscation technique observed in real spam images, and by a degree of obfuscation which can be fine tuned. In particular, we focused on the three kinds of obfuscation techniques widely used by spammers.

2.1 Implementation of text obfuscation techniques

The first obfuscation technique (see Figure 2, top row) consists of making both the text and background colours non-uniform. In particular, the colour of each text pixel is chosen randomly, while the background is made up of horizontal segments of random width and one pixel height, whose colour is also chosen randomly. A Gaussian distribution is used for the grey-level value Y of each text pixel. The corresponding RGB values are then generated as:

$$Y = 0.299 * R + 0.587 * G + 0.114 * B \qquad (1)$$

A Gaussian distribution is also used for the width and the colour of each background segment, which is set as described above for text pixels. For our experiments all the parameters of the above-mentioned probability distributions (mean and variance) were set as described in Section 2.2, with the aim of obtaining images similar to the ones observed in real spam emails, with different degrees of text obfuscation with respect to an OCR tool.

The second obfuscation technique consists of misaligning text characters over a non-uniform background made up of random shapes of different random colours (see Figure 2, middle row). Each text character was shifted vertically by a random amount (positive or negative). The perimeter of each background shape was obtained by connecting with straight lines pairs of points chosen randomly, until a given percentage of the image area was covered (different shapes can overlap). The colours of the background shapes were set as described above for the first obfuscation technique.

The third obfuscation technique consists of drawing horizontal segments of random length over a clean text image with uniform background. The segment colour is identical to the background colour: this results in cutting the

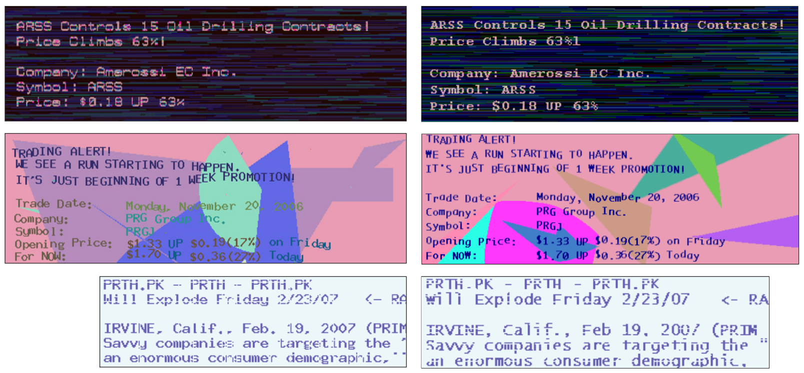
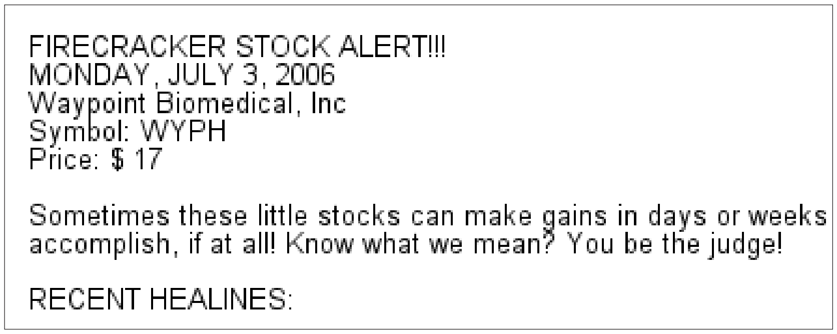

ARSS Controls 15 Oil Drilling Contracts!
Price Climbs 63%!
Company: Amerossi EC Inc.
Symbol: ARSS
Price: $0.18 UP 63%
TRADING ALERT!
WE SEE A RUN STARTING TO HAPPEN.
IT'S JUST BEGINNING OF 1 WEEK PROMOTION!
Trade Date: Monday, November 20, 2006
Company: PRG Group Inc.
Symbol: PRGJ
Opening Price: $1.33 UP $0.19(17%) on Friday
For NOW: $1.70 UP $0.36(27%) Today
PRTH.PK - PRTH - PRTH.PK
Will Explode Friday 2/23/07 <- RA
IRVINE, Calif., Feb. 19, 2007 (PRIM
Savvy companies are targeting the
an enormous consumer demographic,

Figure 2: Examples of real spam images using three different text obfuscation techniques (left), and spam images generated with the proposed algorithm (right). It is easy to see that artificially generated spam images are very similar to the real ones.

text, breaking and splitting characters. The length of each segment, as well as the average horizontal and vertical distance between different segments, is chosen randomly from a Gaussian distribution.

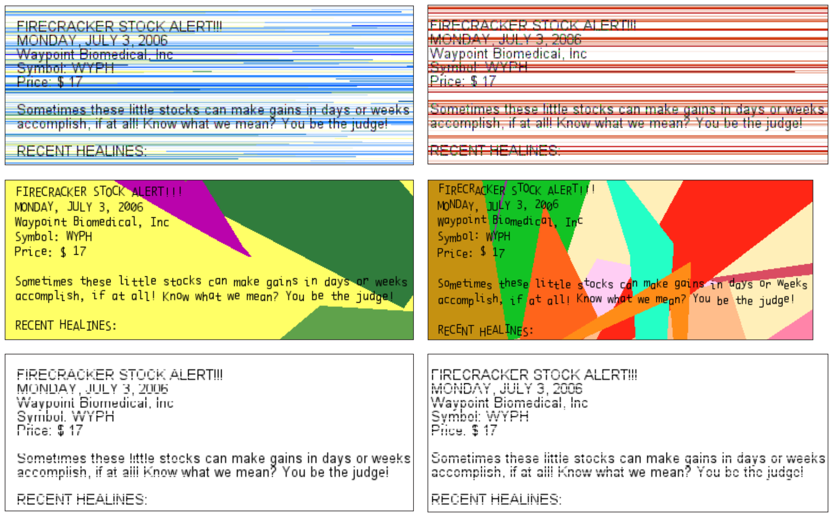


FIRECRACKER STOCK ALERT!!!
MONDAY, JULY 3, 2006
Waypoint Biomedical, Inc
Symbol: WYPH
Price: $ 17

Sometimes these little stocks can make gains in days or weeks
accomplish, if at all! Know what we mean? You be the judge!

RECENT HEALINES:

Figure 3: Example of an image with clean text (degree of obfuscation = 0%).



FIRECRACKER STOCK ALERT!!!
MONDAY, JULY 3, 2006
Waypoint Biomedical, Inc
Symbol: WYPH
Price: $ 17

Sometimes these little stocks can make gains in days or weeks
accomplish, if at all! Know what we mean? You be the judge!

RECENT HEALINES:

Figure 4: Examples of images obtained from the one shown in Figure 3 using the three obfuscation techniques considered in this work, and a degree of obfuscation equal to 50% (left) and 100% (right).

2.2 Definition of the degree of obfuscation

For the purposes of this work, the parameters which control the three obfuscation techniques were set with the following rationale: first, we wanted to obtain images similar to the ones observed in real spam emails. In particular, the text embedded into such images, although obfuscated, must be readable by a human being. Second, a range of parameter values had to be defined to obtain different degrees of text obfuscation with respect to an OCR tool. This was accomplished as follows: we evaluated the performance of the two OCR tools used in *SpamAssassin*, gocr and tesseract, in terms of the word error rate (WER), which is a common measure of OCR performance [10]. For a given image with embedded text, WER is defined as the fraction of words not recognized correctly by the OCR. A word is considered correctly recognized only if all its characters are recognized correctly (in the right sequence).

In these experiments we used a text composed of 80 different words (which is the typical length of an image spam text), excluding punctuation and accents.

For each obfuscation technique we assessed which of the corresponding parameters exhibited a significant correlation with WER. To this aim, we computed WER as a function of each single parameter, setting all the others to constant values (different constant values were evaluated). Afterwards, parameters which turned out not to be significantly correlated to WER were set to constant values so that the images obtained looked as similar as possible to real spam images (without compromising human readability).

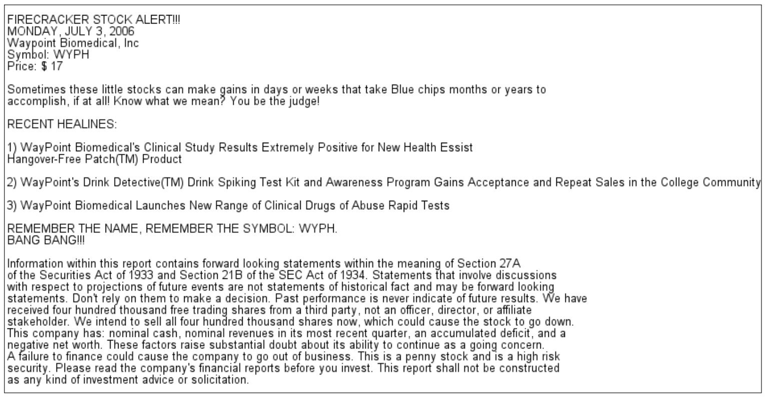
For parameters correlated to WER a range of values was defined so that the values at the end of the range, corresponding to the highest degree of obfuscation, led to images that were still readable by human beings. The degree of obfuscation was then formally defined as a percentage: to obtain an image with, say, a 60% degree of obfuscation, each parameter (among the ones correlated with WER) of the first and second obfuscation techniques was set to $l + 0.6 * (u - l)$, where l and u denote respectively the lower and upper values of the corresponding range (assuming that l corresponds to the lowest obfuscation caused by that parameter). The parameters of the third obfuscation technique were set similarly, but using a logarithmic scale for their range (the reason is that the relationship between the parameters and WER proved to be non-linear for this obfuscation technique).

Examples of the images obtained at different degrees of obfuscation for the image in Figure 3 are shown in Figure 4.

3 EXPERIMENTAL ANALYSIS

In this section we present experiments aimed at evaluating the effectiveness of image spam with obfuscated text in evading detection by the *SpamAssassin* filter equipped with the OCR plug-in.

Two experiments were carried out. In the first experiment, artificial spam images generated as described in Section 2 were used to assess systematically the performance of the single OCR plug-in and of the whole *SpamAssassin* filter as a function of the degree of obfuscation. In the second experiment a real dataset of image spam was used. Version 3.1.3 of *SpamAssassin* was used, with default settings and additional packages related to common collaborative spam-filtering networks, including RAZOR [11], PYZOR [12] and DCC [13]. For the OCR plug-in, the default keywords database was used.



FIRECRACKER STOCK ALERT!!!
MONDAY, JULY 3, 2006
Waypoint Biomedical, Inc
Symbol: WYPH
Price: $ 17

Sometimes these little stocks can make gains in days or weeks that take Blue chips months or years to
accomplish, if at all! Know what we mean? You be the judge!

RECENT HEALINES:

1) WayPoint Biomedical's Clinical Study Results Extremely Positive for New Health Essist
Hangover-Free Patch(TM) Product

2) WayPoint's Drink Detective(TM) Drink Spiking Test Kit and Awareness Program Gains Acceptance and Repeat Sales in the College Community

3) WayPoint Biomedical Launches New Range of Clinical Drugs of Abuse Rapid Tests

REMEMBER THE NAME, REMEMBER THE SYMBOL: WYPH.
BANG BANG!!!

Information within this report contains forward looking statements within the meaning of Section 27A
of the Securities Act of 1933 and Section 21B of the SEC Act of 1934. Statements that involve discussions
with respect to projections of future events are not statements of historical fact and may be forward looking
statements. Don't rely on them to make a decision. Past performance is never indicate of future results. We have
received four hundred thousand free trading shares from a third party, not an officer, director, or affiliate
stakeholder. We intend to sell all four hundred thousand shares now, which could cause the stock to go down.
This company has: nominal cash, nominal revenues in its most recent quarter, an accumulated deficit, and a
negative net worth. These factors raise substantial doubt about its ability to continue as a going concern.
A failure to finance could cause the company to go out of business. This is a penny stock and is a high risk
security. Please read the company's financial reports before you invest. This report shall not be constructed
as any kind of investment advice or solicitation.

Figure 5: Spam image with clean embedded text and several well known spam keywords, e.g. 'stock', 'company', which are in the OCR plug-in keyword list. This image was used as source for generating images with different obfuscation techniques and levels.

3.1 Experiments with artificial spam images

The first experiment we carried out was aimed at assessing the ability of image spam with obfuscated text to evade detection by *SpamAssassin* equipped with the OCR plug-in (gocr was used as the OCR tool).

The analysis was carried out using spam images with different degrees of text obfuscation. To this aim we used a dataset of 1,779 real spam emails with attached images received in the authors' personal mailboxes between July 2006 and February 2007.

To carry out a systematic analysis of the *SpamAssassin* performance as a function of the degree of obfuscation, we substituted the original images attached to each email with artificial images obtained as described in Section 2. More precisely, we first generated an image which was easily detectable as spam by the OCR plug-in: it was a clean image (degree of obfuscation equal to 0%) with embedded text containing several keywords known to be included in the OCR plug-in database, and using a font easily recognized by gocr. The image is shown in Figure 5. This image was attached to each of the 1,779 spam emails to create a dataset with the most favourable conditions for the OCR plug-in, as a baseline for the subsequent comparison with obfuscated images.

Then we modified the image using the three obfuscation techniques described in Section 2, and ten different degrees of obfuscation ranging from 10% to 100% in steps of 10%. For each obfuscation technique and each degree of obfuscation, we generated 1,779 spam images and attached them to the 1,779 spam emails (note that each image was different due to the random choice of the obfuscation parameters).

This led to three datasets, one for each obfuscation technique; each dataset was made up of ten subsets (one for each degree of obfuscation) containing the same 1,779 spam emails, and the emails of each subset contained spam images characterized by the same degree of obfuscation.

We remind the reader that the OCR plug-in has a binary scoring system: it outputs a default value of 3 if the input image is deemed to be spam, and 0 otherwise. Moreover, we point out that even if the OCR plug-in outputs a score value of 3, the input email is not necessarily labelled as spam by *SpamAssassin*, since the final decision also depends on the output of the other modules (which can be negative), and on the fact that the default *SpamAssassin* threshold is 5. Similarly, an email can be labelled as spam by *SpamAssassin* even if the output of the OCR plug-in is 0. For these reasons, in our experiments we evaluated both the performance of the whole *SpamAssassin* filter and the performance of the individual OCR plug-in. Performances were evaluated in terms of the fraction of spam emails correctly labelled as spam, which equals 1 minus the false negative (FN) rate, defined as the fraction of spam emails incorrectly labelled as ham.

Figure 6 shows the OCR plug-in detection rate (1–FN), for each obfuscation technique, as a function of the degree of obfuscation. Note first that at 0% degree of obfuscation, all images were correctly recognized as spam by the OCR plug-in, as desired. It can also be seen that, even if the behaviour of 1–FN depends on the specific obfuscation technique, it almost always decreases as the degree of obfuscation increases (with few exceptions). The decrease is very rapid for the first obfuscation technique, slower for the second one (note that even at 100% degree of obfuscation, about 66% of spam images are still recognized), and abrupt for the third one, once the degree of obfuscation exceeded 80%. Note that at 100% degree of obfuscation (which does not compromise human readability, as explained in Section 2.2) the OCR plug-in was not able to recognize any spam

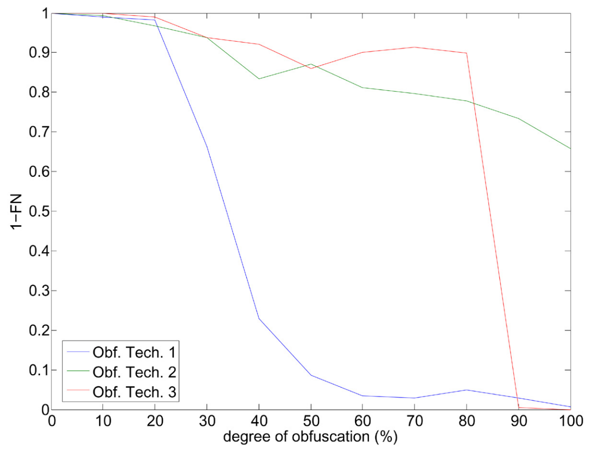


Figure 6: Spam detection rate of the OCR plug-in as a function of the degree of obfuscation for the three obfuscation techniques considered in the experiments.

email when the first and third obfuscation techniques were used. This provides clear evidence that some obfuscation techniques are effective against detection by the OCR plug-in, even with low degrees of obfuscation (this is the case in the first obfuscation technique).

Consider now the performance of the whole *SpamAssassin* filter. In our experiments we analysed its behaviour for each of the four configurations discussed in Section 1. These configurations are denoted in the following as 'bayes' (only the naïve Bayes text classification module was used), 'net' (only Internet-related rules were used), 'bayes + net' (both kinds of rules were used) and 'local' (none of these rules was used).

The naïve Bayes text classifier was trained on 5,273 spam emails collected in the authors' mailboxes (from November 2005 to June 2006) and 3,515 ham emails taken from the *Enron* dataset [14, 15] (since the naïve Bayes is a statistical classifier, we chose a 2:3 proportion between ham and spam training emails, which, according to recent estimates, is similar to the proportion observed in real email traffic).

In Figure 7 we report the 1–FN values as a function of the degree of obfuscation for each of the four configurations and each of the three obfuscation techniques. First, as one might expect, it can be seen that the 'bayes + net' configuration is the most effective, while 'local' is the least effective. Interestingly, the use of Internet-related rules only ('net') significantly outperformed the use of the naïve Bayes module only ('bayes').

Consider now the performance of *SpamAssassin* with respect to the obfuscation techniques: it is easy to see that the following features are similar to the behaviour of the OCR plug-in. The rate of correctly recognized spam emails almost always decreases for increasing degrees of obfuscation. Such a decrease is smooth for the first and second obfuscation techniques, while it is abrupt for the third technique. Moreover, the second technique proved to be less effective in evading *SpamAssassin* (since it results in higher 1–FN values at 100% degree of obfuscation).

However, a remarkable difference appears with respect to the OCR plug-in performance: even at 100% degree of

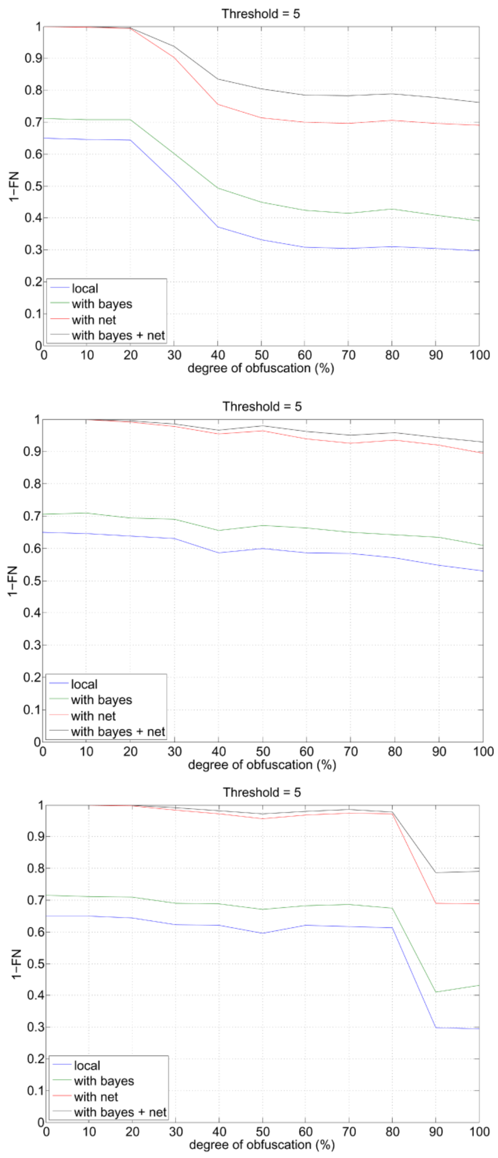


Figure 7: SpamAssassin detection rate versus the degree of obfuscation, for obfuscation techniques 1 (top), 2 (middle) and 3 (bottom), and the four different SpamAssassin configurations, using the default threshold value 5.

obfuscation a significant fraction of spam emails is correctly recognized by *SpamAssassin* when the first and third obfuscation techniques are used, although in these conditions the OCR plug-in labelled all the images as ham (see Figure 6). This means that a large fraction of image spam (between 75% and 90% in our experiments across the different obfuscation techniques when the 'bayes + net' configuration was used), can be recognized even if the OCR plug-in is evaded, thanks to the other filtering rules. Instead, the remaining fraction of image spam emails (from 10% to 25%, depending on the obfuscation technique) can be detected by *SpamAssassin* only thanks to the OCR plug-in (the lower the degree of obfuscation, the higher the detection rate).

To sum up, our experimental results suggest that *SpamAssassin* is rather robust against image spam (perhaps more robust than one might think), even if its OCR-based filtering module can be evaded quite easily using obfuscated text. Using an OCR-based filtering module can improve *SpamAssassin*'s detection capability further if text embedded into images is clean or exhibits very low degrees of obfuscation.

3.2 Experiments with real spam images

In this section we provide an evaluation of *SpamAssassin* on a real dataset of image spam, in order to assess its performance in a realistic working environment. In this case we compare the performance of *SpamAssassin* with and without the OCR plug-in. We predicted that the improvement in spam detection rate due to the use of an OCR-based plug-in would be lower than the maximum one observed in the previous experiments (corresponding to clean spam images), given that many real spam images contain obfuscated text. For these experiments we used the same emails as in the previous experiments plus 253 emails with more than one attached image.

In Table 1, rows (a) and (b), we report the 1–FN values of *SpamAssassin* respectively with and without using the OCR plug-in for the four configurations explained above. Focusing on the most effective configuration ('bayes + net', fourth column), it can be seen that over 80% of image spam emails were recognized by *SpamAssassin* without the OCR plug-in. When the OCR plug-in was used, only 6% more image spam emails were detected (this percentage is reported in row (c)).

To analyse these results further, in row (d) we report the percentage of emails correctly labelled as spam by both the OCR plug-in and *SpamAssassin*, and in row (e) the percentage of spam emails correctly labelled as spam by the OCR plug-in and mislabelled as ham by *SpamAssassin*. It can be seen that the OCR plug-in detects only about 25% of image spam (and, as shown in row (c), only for 6% of spam emails was such detection useful for the overall *SpamAssassin* filter). As shown by row (e), the detection of a spam image by the OCR plug-in was always sufficient for it to be labelled correctly as spam by *SpamAssassin*.

	local	net	bayes	bayes+net
(a)	41.3%	51.4%	78.4%	86.8%
(b)	31.6%	46.2%	70.3%	80.8%
(c)	9.7%	5.2%	8.1%	6.0%
(d)	15.8%	15.9%	25.3%	25.3%
(e)	9.5%	9.4%	0%	0%

Table 1: Percentages of spam emails detected by SpamAssassin using the OCR plug-in (a); spam emails detected without using the OCR plug-in (b); spam emails correctly labelled by SpamAssassin only when the OCR plug-in was used (c); spam emails labelled as spam both by the OCR plug-in and SpamAssassin (d); spam emails labelled as spam only by the OCR plug-in (e).

The above results seem to confirm that most image spam can be detected by *SpamAssassin* even without using an OCR-based filtering module (that is, they are recognized for characteristics other than the text embedded into images). Nevertheless, there is also evidence that such kinds of module can be useful against those kinds of image spam in which text obfuscation techniques are less likely to be used (e.g. phishing emails, which, to be effective, must look as if they come from reputable senders, and thus should be as 'clean' as possible).

It should also be noted that a higher effectiveness than that exhibited by the OCR plug-in could be attained by using more effective OCR tools (although this could lead to an undesirable higher computational complexity), perhaps tailoring them to the characteristics of image spam (mainly low-resolution images), and exploiting different text analysis techniques from the simple keyword detection carried out by the OCR plug-in.

However, we believe that different techniques should be used to detect the percentage of image spam with obfuscated text which currently evades a filter like *SpamAssassin* (about 15% in our experiments on real spam emails). For instance, techniques based on pattern recognition and computer vision techniques can be used, like the ones in [16, 17], and the ones investigated by the authors in [18–20], which are aimed specifically at detecting the presence of obfuscated text. A combination of such kinds of approach and OCR-based approaches could also allow a tradeoff between effectiveness and efficiency. A hierarchical architecture can be devised for a spam filter, in which computationally demanding OCR tools are used only if the email has not been recognized as spam by other techniques.

4 CONCLUSIONS

We assessed the performance of *SpamAssassin* equipped with the OCR plug-in against image spam with obfuscated text. We used an artificial image spam generator to simulate three different obfuscation techniques and generate text images with different degrees of obfuscation. It was found that, although the OCR plug-in is effecive on clean images, it can be evaded quite easily using obfuscated text. Nevertheless, *SpamAssassin* turned out to be rather robust against image spam thanks to the other filtering rules: a large fraction of image spam was detected for characteristics other than the text embedded into images. Similar results were obtained from experiments carried out on a real image spam stream.

These results suggest that OCR-based filtering modules can be useful only against those kinds of image spam, like phishing, in which text obfuscation techniques are not likely to be used. For obfuscated image spam, we advocate the use of approaches which take into account explicitly the adversarial spammer's actions, namely, approaches which recognize spam images by detecting the presence of obscured text. Our recent works show that this approach is more suitable to detect image spam with obfuscated text [18–20]. In particular, we believe that the most effective solution will be provided by a combination of these approaches, possibly arranged in a hierarchical architecture to limit the drawback of the high computational cost of OCR tools.

REFERENCES

[1] Sahami, M.; Dumais, S.; Heckerman, D.; Horvitz, E. A Bayesian approach to filtering junk e-mail. AAAI Technical Report WS-98-05, Madison, Wisconsin, 1998.

[2] Drucker, H.; Wu, D.; Vapnik, V. N. Support vector machines for spam categorization. IEEE Transactions on Neural Networks, 10(5):1048–1054, 1999.

[3] Graham, P. A plan for spam, 2002. http://paulgraham.com/spam.html.

[4] Androutsopoulos, A.; Koutsias, J.; Cbandrinos, K. V.; Spyropoulos, C. D. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal email messages. Proceedings of the ACM International Conference on Research and Developments in Information Retrieval, pp.160–167, 2000.

[5] http://spamassassin.apache.org/.

[6] Available at http://jocr.sourceforge.net/.

[7] Available at http://code.google.com/p/tesseract-ocr/.

[8] http://wiki.apache.org/spamassassin/MassCheck.

[9] Fumera, G.; Pillai, I.; Roli, F. Spam filtering based on the analysis of text information embedded into images. Journal of Machine Learning Research (special issue on Machine Learning in Computer Security), 7:2699–2720, 2006.

[10] Vinciarelli, A. Noisy text categorization. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(12):1882–1885, 2005.

[11] http://razor.sourceforge.net/.

[12] http://pyzor.sourceforge.net/.

[13] http://www.rhyolite.com/anti-spam/dcc/.

[14] Available at http://www.cs.cmu.edu/~enron/.

[15] Klimt, B.; Yang, Y. The Enron corpus: a new dataset for e-mail classification research. Proceedings of the European Conference on Machine Learning, pp.217–226, 2004.

[16] Dredze, M.; Gevaryahu, R.; Elias-Bachrach, A. Learning fast classifiers for image spam. Proceedings of the International Conference on Email and Anti-Spam (CEAS 2007).

[17] Wang, Z.; Josephson, W.; Lv, Q.; Charikar, M.; Li, K. Filtering image spam with near-duplicate detection. Proceedings of the International Conference on Email and Anti-Spam (CEAS 2007).

[18] Biggio, B.; Fumera, G.; Pillai, I.; Roli, F. Image spam filtering by content obscuring detection. Proceedings of the International Conference on Email and Anti-Spam (CEAS 2007).

[19] Biggio, B.; Fumera, G.; Pillai, I.; Roli, F. Image spam filtering using visual information. Proceedings of the International Conference on Image Analysis and Processing (ICIAP 2007), IEEE Computer Society, pp.105–110.

[20] Biggio, B.; Fumera, G.; Pillai, I.; Roli, F. Image spam filtering using textual and visual information. Proceedings of the MIT Spam Conference 2007.

[21] McCallum, A.; Nigam, K. A comparison of event models for Naive Bayes text classification. Proceedings of the AAAI Workshop on learning for text categorization, 1998.