MARCH 2011

virus BULLETIN

Fighting malware and spam

CONTENTS

ISSN 1749-7027

IN THIS ISSUE

EVOLUTION THEORY

Drawing on some parallels with molecular biology, the W32/Flibi virus attempts to evolve new behaviours in order to evade detection. Peter Ferrie has the details.

page 4

UPWARDLY MOBILE

Until recently, malware on mobile devices had not been used for organized crime involving large amounts of money. This changed when the infamous Zeus gang started to show a clear interest in infecting mobile phones. Axelle Apvrille and Kyle Yang present an in-depth analysis of the Zitmo trojan.

page 6

VBSPAM CERTIFICATION

In this month's VBSpam test 18 out of 19 full solutions achieved VBSpam certification. Martijn Grooten has the details.

page 16

'Such products could dupe even the most experienced of users.'

Dmitry Bestuzhev
Kaspersky Lab

WHAT NEXT FOR ROGUE AVS?

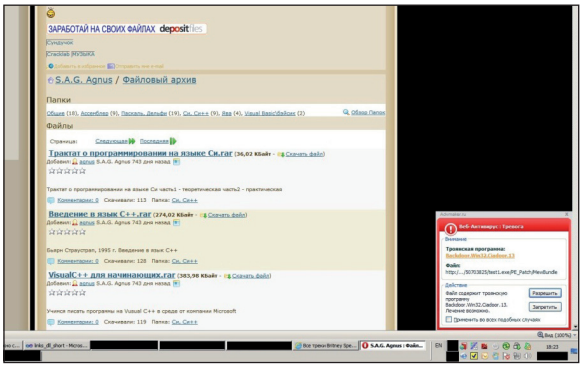
In 2009 I wrote a blogpost about a new trend in rogue AV attacks[1] – the cloning or copying of the GUIs of genuine anti-malware products. I predicted that we would soon see rogue AVs that were visually almost exact copies of legitimate security software – and only a few months later, we saw some very convincing imitations: fake *Kaspersky*, *Avast*, *Symantec*, *McAfee* and *Avira* products.

In May 2010, we saw a new rogue AV attack featuring a fake version of *Microsoft Security Essentials*, and at the beginning of 2011 we saw a whole new wave of rogue AVs that included fake versions of *AVG 2011* and *Kaspersky Anti-Virus for Windows Workstations*. Our predictions couldn't have been more accurate. However, in addition to the abovementioned, I recently found an 'interesting' website. Try to spot what's wrong in the screen shot at the top of the next column.

The answer lies in the bottom right-hand corner. The criminals behind rogue AV attacks now not only clone GUIs, but also copy attack notifications as well. The red balloon you see is an exact copy of a standard *KAV for Workstation* malware detection notification.

So, what is the main problem here and what will it lead to? Well, in 2009 alone, the FBI estimated[2] that victims of rogue AV attacks lost around US$150 million, and I firmly believe that the figure for 2010 will be higher.

[1] http://www.securelist.com/en/blog/208187938/Rogue_AV_raising_the_stakes
[2] http://www.ic3.gov/media/2009/091211.aspx



In most cases, victims lost their money after having been tricked into paying for purported disinfections after receiving fake notifications. However, today the picture is changing. The issue facing even the most experienced users is how to distinguish between fake and legitimate anti-virus solutions before installing them on a computer. Just imagine a rogue AV being installed that has cloned virtually every aspect of the original solution: from the GUI to the names of malware. Maybe a handful of advanced users could check the system's process names and file paths to determine the legitimacy of the software, but even this can be faked by the criminals. They can create the same paths, the same file names and the same process names. So, what we are facing here is the likelihood that criminals will not only make money from victims by employing fake virus warnings, but also from the sale of rogue AV products. As mentioned previously, such products could dupe even the most experienced of users.

The criminals have another advantage too. Many legitimate AV companies not only sell their goods via an official website, but also through online stores and partner sites. What if one or more of these sites were hacked and a new, high-tech rogue AV delivered to a user's machine? Who would pay for the damage – the online store or the vendor?

Now imagine another scenario: what if the criminals start registering and building fake online stores selling several cloned AVs? How would the average user be able to tell if they were purchasing a real or a fake solution?

The matter could partially be resolved if vendors were to sign files so that users could check the digital signature. However, we know from recent experience that digital certificates are not an absolute guarantee that a file is legitimate and clean. This would also require users to be experienced enough to know how to check signed files.

With no ideal method of protecting users against these threats, the AV industry needs to generate some new ideas as to how we can stamp out this type of fraud.

NEWS

CHINA STARTS TO CLEAN UP ITS ACT

Once the number one source of spam in the world, China appears to be slipping down the list of the world's top spam-producing countries.

While just two years ago the country consistently featured in lists of the top five spam-producing countries, China is currently ranked at number 18 by *Cisco Systems*' *IronPort* group and at number 20 by *Sophos*.

There are a number of factors that may have contributed to the decrease in spam coming from the country. China launched an anti-spam initiative in 2006 that brought together network operators and security companies to address the problem, but the most dramatic drop in spam levels wasn't seen until towards the end of 2009. There is also an ongoing joint initiative between US spam fighters and the Internet Society of China which aims to set standards and determine ways in which the two countries can cooperate on tackling the problem (a report is due out on the project's progress later this month). According to *Cisco*, ISPs in China have become better at working with customers to reduce the spam problem, and the country's new, tougher regulations for registering Internet domains and controls over who is allowed to send email may also have helped bring spam volumes down.

While this is good news for China, it appears that many of the spammers who had been sending their emails from China have simply moved their operations to other countries (for example Russia), where Internet controls and anti-spam regulations are slack enough to allow them to continue.

MESSAGING ANTI-ABUSE AWARD

In memory of MAAWG supporter Mary Litynski, the Messaging Anti-Abuse Working Group has created an annual award to recognize an individual who has made a significant contribution to making the Internet safer for all. Through the award, MAAWG seeks to bring attention to the behind-the-scenes work undertaken by dedicated and driven individuals to help reduce spam, fight bots or malware, or safeguard other aspects of online messaging.

The first MAAWG Mary Litynski Award will be presented at the group's annual San Francisco meeting in June 2011. Nominations can be made online at http://www.maawg.org/events/maawg-mary-litynski-award.

HACKERS HACKED

The tables were turned last month when a forum used by major players in the Russian cybercrime community found itself the victim of an attack, and details of over 2,000 of its members were leaked. According to Russian website *Life News*, active members of the forum included prolific spammers, carders and malware distributors.

Prevalence Table – January 2011 [1]

Malware	Type	%
Autorun	Worm	9.73%
Conficker/Downadup	Worm	6.33%
VBinject	Trojan	5.31%
Agent	Trojan	3.97%
VB	Worm	3.86%
Downloader-misc	Trojan	3.76%
FakeAlert/Renos	Rogue AV	3.56%
OnlineGames	Trojan	3.34%
Heuristic/generic	Virus/worm	3.26%
Injector	Trojan	3.15%
Delf	Trojan	2.87%
Adware-misc	Adware	2.66%
Exploit-misc	Exploit	2.02%
Crack/Keygen	PU	1.98%
Sality	Virus	1.94%
Zbot	Trojan	1.77%
Heuristic/generic	Trojan	1.75%
Small	Trojan	1.72%
AutoIt	Trojan	1.63%
Crypt	Trojan	1.52%
Virtumonde/Vundo	Trojan	1.45%
Hupigon	Trojan	1.40%
Virut	Virus	1.39%
Zwangi	Adware	1.39%
Bifrose/Pakes	Trojan	1.34%
Dropper-misc	Trojan	1.34%
Alureon/Olmarik	Trojan	1.20%
Bancos	Trojan	1.09%
Iframe	Exploit	1.05%
Tanatos	Worm	1.04%
StartPage	Trojan	1.01%
Potentially Unwanted-misc	PU	0.96%
Others [2]		19.18%
Total		100.00%

[1] Figures compiled from desktop-level detections.

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

MALWARE ANALYSIS

FLIBI NIGHT

Peter Ferrie
Microsoft, USA

If we were to consider a computer virus to be a life form, then we could perhaps extend the analogy to include predators such as observant users and anti-malware solutions. We could also consider the need for mutation to produce new behaviours in order to evade predators and survive. The W32/Flibi virus aims to do just that.

EVOLUTION OF EVOLUTION

Previous efforts at evolving software have generally been in the form of fully programmed traits that are chosen using a weighting system, so that some traits are favoured more than others. An example of a piece of malware using this technique is W32/Simile (see *VB*, May 2002, p.4). This is a kind of evolution, but it cannot produce new behaviours. A variation on that theme was used by W32/Zellome (see *VB*, May 2005, p.7), whose traits corresponded to functions for the polymorphic decryptor, rather than the virus code itself. This cannot produce new behaviours either.

Flibi takes a new approach, and has some parallels with molecular biology. Each byte of the virus code (apart from the translator function) is equivalent to a codon[1]. We have a single 'reading frame', so each codon can be read in only one way, no matter where one starts reading. The translator function is equivalent to tRNA[2]. Its purpose is to convert the codons into amino acids[3]. Unlike in molecular biology, however, Flibi has no start and stop codons[4]. Instead, the translator function knows where the codon chain starts and how long it is (though both of these values are vulnerable to mutation which can affect future generations).

tRNA AT WORK

The virus begins by allocating 64KB of memory for the 'organism'. The preview version of the virus allocates the region using only read/write attributes. As a result, that version will not run if Data Execution Prevention (DEP) is enabled for all processes, but the bug was fixed in the release version of the virus. The translator function then converts the codons into amino acids and places them in the memory block. This is achieved by using the codons as an index into a table of instruction blocks. Each codon corresponds to one instruction block, and all but two of the blocks contain only a single instruction (the two exceptions contain two instructions each). There are many redundancies in the table (that is, many codons map to the same instruction). This is intentional, since it increases the robustness of the overall organism. Minor mutations can allow a modified codon to continue to be translated to the same amino acid (and this is yet another parallel to molecular biology). Each instruction block is eight bytes long, which leaves room for further minor mutations to occur without affecting the instruction itself.

IMPORT-ANT ASSIGNMENT

The virus builds the strings 'kernel32.dll' and 'advapi32.dll' using a sequence of add instructions. There are some other ways to achieve this, but the language contains only 45 instructions in the release version of the virus (43 instructions in the preview version), so there are not many other ways. (As a side note, the language can be simplified to as few as 18 instructions. The details will be provided in a future article.)

The virus loads kernel32.dll and then builds a series of pointers to memory, using another sequence of add instructions. The virus stores the image base value, and then locates the import table. It copies the import table pointers to a local buffer in memory. The virus parses the import table directly in order to resolve the APIs that it needs to replicate. Unlike other viruses that avoid using the GetProcAddress() API so that the import list is harder to determine, this virus avoids using the GetProcAddress() API in order to cause the list of APIs to become vulnerable to mutation. The virus still uses a list of hashes instead of function names, but if any of those hashes have been altered by mutation, then an entirely different API address might be retrieved, resulting in completely different behaviour. This is just one place where a new behaviour might arise from 'evolution'.

The virus parses the import table but in a rather peculiar way. Instead of examining the characters of the API name until the end of the name, the virus reads the first ten characters regardless of the length of the name. The characters are hashed using add, sub and xor, before and-ing the result to isolate the low 12 bits. There are three problems with this approach. One is that the API name

[1] A codon is a trinucleotide sequence of DNA or RNA (the nucleic acids that contain the genetic instructions used in the development and functioning of living organisms) that corresponds to a specific amino acid. See http://www.genome.gov/Glossary/index.cfm?id=36.

[2] Transfer RNA, or tRNA, is a small RNA molecule that is involved in protein synthesis. See http://www.wiley.com/college/boyer/0470003790/structure/tRNA/trna_intro.htm.

[3] Amino acids are the building blocks of proteins. See http://en.wikipedia.org/w/index.php?title=Amino_acid&oldid=412676887.

[4] See http://en.wikipedia.org/w/index.php?title=Genetic_code&oldid=412677908#Start.2Fstop_codons.

might be longer than ten characters, resulting in a possibly false match if those characters are shared by another API.

The second problem is that the API name might be shorter than ten characters, resulting in characters from the following name being read and incorporated into the hash. In that case, if other versions of *Windows* have different APIs after the one of interest, then the hash will not match on that platform. In fact, that is exactly what happened for the Sleep() API. The preview version of the virus carries a hash for the Sleep() API that is suitable for *Windows 2000* or *Windows XP*, where the Sleep() API is followed by the SleepEx() API. However, on *Windows Vista*, two additional APIs were inserted between the Sleep() and SleepEx() APIs: SleepConditionVariableCS() and SleepConditionVariableSRW(). The result is that the preview version of the virus crashes when run on *Windows Vista* or *Windows 7*. This bug was fixed in the release version by importing the Sleep() function explicitly.

The third problem is that the hashing algorithm is very weak and can easily result in false matches. It is unclear why the virus author did not use any kind of arithmetic rotation, given that the language supports both shift left and shift right operations. These operations can be combined to perform an arithmetic rotation in either direction.

After resolving the API addresses from kernel32.dll, the virus loads advapi32.dll and then branches to the code above while attempting to resolve some APIs from that DLL. After resolving the API addresses from advapi32.dll, the virus loads advapi32.dll again, because the API resolver code is not a subroutine, so the same code path is reached for the second time. However, the virus recognizes this case and it does not resolve the API addresses again.

WHIP IT INTO SHAPE

The virus constructs a new filename consisting of eight randomly chosen lower case letters, and then appends '.exe'. This is the name of the next-generation file. The virus retrieves the command line that was used to launch it. If the first character in the command line is a double quote, then the virus skips it and searches for the last double quote. Everything in between is extracted for use. If the command line does not begin with a double quote, then it is used as it is. The virus copies itself as 'x:\evorisss.exe', where 'x' is the drive letter taken from the command line. It also copies itself to the next-generation filename.

The virus opens the next-generation file and maps it into memory. For each byte in the file, there is a 0.02% chance that one of its bits will be flipped. There is also a 0.003% chance that any two adjacent dwords will be exchanged. Finally, there is a 20% chance per execution of the virus that between one and 32 bytes will be deleted by being overwritten with a corresponding number of the bytes immediately prior to the selected block, and then overwriting the prior bytes with no-operation instructions.

These mutations are often lethal, since the file headers are vulnerable to alteration, resulting in invalid executables. The random deletion of instructions will also eventually result in code that does the absolute minimum (such as copying itself only to the root directory of the current drive, and requiring user interaction in order for it to run again) in order to retain the 'worm' characteristic.

PAYLOAD

The virus carries a payload that displays with a 12.5% chance per execution of the virus. The payload is to display the message 'Kadyrov is a murderer!!!'. There are several 'infamous' people with the name Kadyrov; it is not known to whom this message refers. The string is not visible in the virus code, because it is constructed using yet another sequence of add instructions.

RUN WORM RUN

The virus waits for between 0 and 15ms, and then creates the registry key 'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run'. The virus sets the default value to 'x:\evorisss.exe', where 'x' is the drive letter taken from the command line, as above. The virus also creates an autorun.inf file in the current directory, containing a reference to the next-generation file. The idea is that by browsing to the directory that contains autorun.inf, the worm will be launched without further user interaction.

The virus also enumerates all drives and checks the type of each of them. If the drive is removable, fixed, remote, or a ramdisk, then the virus attempts to copy autorun.inf to that drive, along with the next-generation file. There is a bug in this routine in the preview version of the virus, which is that if the drive is a floppy drive and there is no disk in it, then *Windows* will display an error message. This bug was fixed in the release version. The virus repeats the enumeration approximately once every seven minutes.

CONCLUSION

Evolution in software. It's an interesting idea, but this life form's biggest challenge is simply to survive. If an entire race of dinosaurs can be wiped out by a single meteor, it seems likely that good generic detection by anti-malware software could perform the same feat and kill this worm before it ever gets a chance to mutate beyond recognition.

TECHNICAL FEATURE

DEFEATING mTANS FOR PROFIT – PART ONE

Axelle Apvrille, Kyle Yang
Fortinet

Malware on mobile phones has existed for several years, but until recently it had not been used for organized crime involving large amounts of money. This changed in September 2010 when the infamous Zeus gang, known for targeting online banking, started to show a clear interest in infecting mobile phones and released a new version of their bot to propagate a trojan for mobile phones.

In this two-part article (based on a paper presented at ShmooCon 2011) we will present an in-depth reverse engineering of the mobile phone trojan, show how to reroute stolen SMS messages to a test phone, and explain how to display hidden windows of the trojan.

1. INTRODUCTION

For years, people have been predicting that malware for mobile phones will one day follow the same path as computer viruses and become both a significant problem and part of major cybercriminal activities [1–4]. There are a number of reasons for this prediction: first, the number of mobile phones now outnumbers the number of desktop or laptop computers, and so a single mobile sample has the opportunity to spread widely. The recent *Symbian* Yxes worm, for instance, has been reported to have infected hundreds of thousands of Chinese subscribers [5]. Second, mobile phones make a perfect target for attackers, because, on the one hand, they are being used for sensitive operations (e.g. payments, storage of personal data) and on the other, they are quite easy to attack because they offer many communication interfaces, with little user education and weak protection. Third, as the gap between mobile phones and personal computers (laptops, netbooks etc.) narrows, mobile malware is expected to evolve quickly, inheriting skills from the past history and experience of computer viruses.

As an example, it took 20 years for the first polymorphic virus (V2PX) to appear on computers, whereas it took only eight years for one to appear on mobile phones (WinCE/Pmcryptic.A!tr).

Fortunately, the much feared mobile Death Star has not yet hit. Significant steps towards the dark side have nonetheless been noticed. In the old days, pre-2004, mobile viruses were mostly the product of enthusiast hackers curious to understand how their devices worked. But 2004 proved to be a turning point, with the appearance of several new proof-of-concepts including Duts, Brador and Cabir [6].

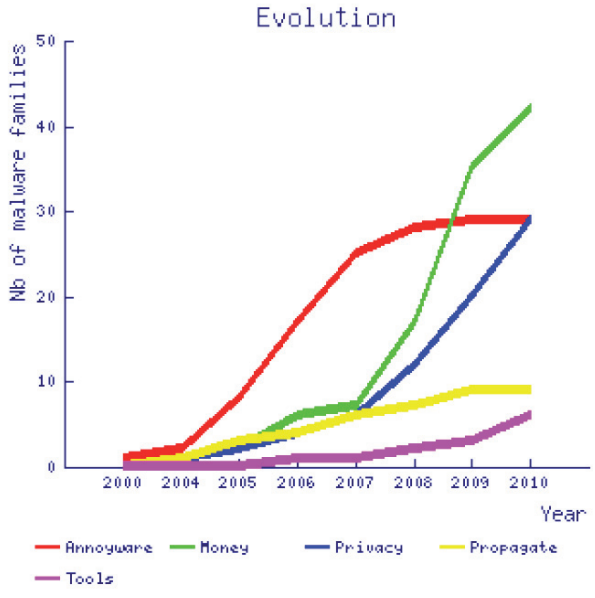


Figure 1: Number of new malware families identified for a given category and per estimated year of implementation.

These opened the door for a new wave of mobile viruses, most of which were annoyware (viruses that disabled fonts, rebooted the phone etc.) for *Symbian* phones. By 2009, mobile malware authors had started to show more interest in making some quick money [7] – for example with diallers which called international numbers. This change of mood was confirmed in 2010, with *Fortinet*'s internal statistics showing an increase in monetized malware (approximately 30% in 2010) and malware targeting privacy (25%), while no new annoyware was detected (see Figure 1). This seemed to confirm that mobile malware is moving into the realms of larger-scale cybercriminality.

However, in most cases, the rewards attackers reap remain rather low[1]: interesting for an individual attacker, but not significant enough to be interesting to organized crime. For example, the authors of the *Windows Mobile* Terdial dialler generated 135 euros in the first few days it spread. This may be worthwhile for a single attacker, but it is not enough to interest a large cybercriminal group. It is not easy for malicious diallers to generate more substantial revenue because premium phone numbers are rented for short periods and the revenue is divided between all resellers.

Zitmo is the first sign of organized criminals showing an interest in mobile malware. It is being propagated by newer versions of the infamous Zeus botnet which has a

[1] (Although, due to the fact that victims and operators provide little information regarding the attacks they encounter, it is difficult to evaluate true financial losses.)

record of attacking online banking. Zitmo, short for 'Zeus In The MObile', infects mobile phones and intercepts SMS messages carrying a one-time password known as a mobile Transaction Authentication Number (mTAN). mTANs are commonly used by banks as an additional way to secure authentication of sensitive online transfers or transactions.

Zitmo was first discovered by *s21sec* [8] and shortly after by *Fortinet* [9], and a new variant was identified in February 2011. It has been written about in several blog posts and press releases, but as usual the full story has not been disclosed, or else is too fragmented in various places for readers to make the connections. This article is about making those connections so as to help understand how the attack works from start to end, but also revealing unknown aspects discovered during our research, such as similarities with other commercial spyware and a hidden debug window.

We will start by providing some background information on both Zeus and mobile malware. Then, we explain how the attack behind Zitmo works, from cybercriminal to bank account. We also cover the presumed making of Zitmo. In the second instalment of the article (next month) we will present our reverse engineering work of Zitmo, explaining each feature of Zitmo (startup, SMS interception, commands and corresponding actions). Finally, we will attempt to draw lessons from the attack, highlighting the roots of the attack and ideas to circumvent it.

2. BACKGROUND AND RELATED WORK

In this section we provide some background information that will be useful to know before getting into the details of the specific Zeus/Zitmo attack.

Zeus (also known as Zbot) is a crimeware kit which allows a botnet herder to configure and create custom executables that precisely target their intended hosts [10]. The kit is primarily designed for stealing banking information, although it can easily be used for other types of data or identity theft. As it is sold in the underground market, there are several different botnets – each one targeting the specific hosts its bot herder configured – rather than there being a single Zeus botnet. For example, in the case of Zitmo, the configuration downloaded by the specific Zbot executables targeted several Spanish online banking and financial organizations.

The Zbot executable works as follows: first, it decrypts a hard-coded XOR-encrypted BLOB, which contains two pieces of information:

1. A URL, from which to download an encrypted configuration file.
2. A table, which corresponds to a scheduled key for the RC4 algorithm.

Next, it downloads a configuration file from the URL. The configuration file is both encrypted and compressed. The binary decrypts the file – as the RC4 key is already in its scheduled format, RC4 decryption boils down to processing it in RC4's pseudo random generation algorithm (PRGA) [11]. Finally, the binary XORs the result with the initial encrypted configuration file to retrieve a compressed configuration file.

After decompression, the configuration file is parsed by the bot [12] in order to inject JavaScripts into bank pages. Then, the bot sends a report back to the server and waits for further commands. Figure 2 shows a few of the URLs targeted by the botnet and into which it injects JavaScript. Zitmo contained a form to get the victim's phone number and model [8].

From the point of view of mobile malware, it is important to note that although the *iPhone*, *BlackBerry* and *Android* are making an impression in the market, *Symbian* was still the most widespread mobile platform in 2010 (41% according to *Gartner*). In particular, Zitmo targets phones running *Symbian OS* version 9.1 or greater. The malware authors also planned to release a version for *BlackBerry*, but apparently never did so. *Symbian OS 9.1* was released in 2005. It includes security features such as data caging and capabilities. In 2009, Yxes was one of the first pieces of malware to appear on this platform [13].

There are still relatively few tools and documents available to reverse engineer and experiment with *Symbian OS 9* – security researchers are usually more attracted by research on the *iPhone* or *Android* phones. Among our favourite tools, we would cite *IDA Pro* (which supports ARM assembly code), *SISWare* for unpacking, and *App TRK* for remote debugging. The use of those tools is illustrated in [13–15], which contain valuable information regarding the reverse engineering of those phones.

3. ZITMO IN A NUTSHELL

This section is intended to provide an overview of how Zitmo works, and serve as a reference for the rest of the

```
[WEBINJECTS]
...
0030 https://[REMOVED]anesto.es/BANESNETEMPRESAS3/ ...
0031 https://[REMOVED]tander.es/bog/sbi*
0032 https://[REMOVED]tander.es/bog/sbi?ptns=refres&paso=*
...
```

Figure 2: Part of the configuration file which matches the Zitmo attack. (Courtesy of David Barroso.)

article. Mainly, it puts together and clarifies information found in [8, 16] and our prior work [17, 18], in addition to explaining facts which have not explicitly been highlighted before.

3.1 Two-factor authentication for online banking

To secure online banking transactions, sensitive operations or logins, several banks (e.g. *ING Direct*, *BB Bank*, *Deutsche Postbank* – mostly European banks) use two-factor authentication. Two-factor authentication means that two different authentication methods must be combined from three possible choices: what the end-user knows (e.g. a password), what he has (e.g. a physical token) and what he is (e.g. biometric data). For example, to perform a sensitive operation on an account, banks typically ask the end-user to enter a valid identifier and password (something he knows) and a secret one-time PIN sent to him by SMS (something he has). This secret code sent by SMS is also known as an mTAN – a mobile Transaction Authentication Number. This solution is believed to be quite secure [19]. Unfortunately, Zitmo demonstrates a practical way to defeat it.

3.2 Infection process

The attack works as follows: first, the victim's computer is infected by a sample of Zeus (see Section 2). The compromised computer becomes part of a Zeus botnet, controlled by a botmaster (see Figure 3, step 1). The next time the end-user visits his favourite bank's website (e.g. *Santander Bank* for Zitmo), the infected computer lets the HTTP request go, but modifies the content of the response on the fly and dynamically injects a malicious script into the bank's HTML code (step 2). The victim's browser displays the modified content – which is controlled by Zeus. Note that this is not the same as phishing: the URL displayed in the browser is real and corresponds to the legitimate bank, whereas, in phishing, the URL corresponds to a website controlled by the attackers. The modification of content does not consist of pointing to another website, but of code injection directly on the compromised computer of the victim.

In the case of Zitmo, the injected script builds an HTML form which requests the victim's mobile phone number and model. These details are purported to be important information required 'for security'. The end-user has no reason to be suspicious as he believes the page he is viewing is legitimate. Of course, the phone number and phone model are not sent to the bank, but transferred to the bot master. The bot master contacts another host with SMS sending capabilities, which sends an SMS to the victim's phone. This SMS claims to be a digital security certificate, and provides a

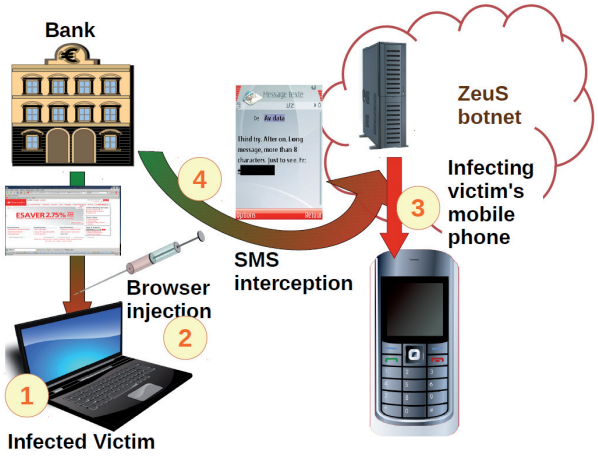


Figure 3: Infection by Zitmo and intercepting the mTAN.

link to download the fake certificate (step 3). The download backend was hosted on krambamburr.com (fast flux).

Again, the victim has no reason to be suspicious because he believes the SMS has come from the bank. Note that SMSs are a handy way to propagate malware because, unlike MMS, all mobile devices are generally properly configured to receive them.

The end-user downloads the alleged security certificate (cert.sis) from a remote website. At this point, a knowledgeable end-user might become suspicious, because certificates are usually sent using the PKCS#12 format (extension .p12 or .pfx), rather than using a *Symbian* installation package (.sis or .sisx extension). In reality, cert.sis is not a certificate, but a modified version of a known Russian spyware program named SMS Monitor [16].

Finally, the end-user installs the trojan on the phone (believing it to be the security certificate). Now both authentication vectors – the victim's computer and his phone – are infected.

3.3 mTAN hijacking

The trojan consists of a malicious daemon, named NokiaUpdate.exe, which listens to incoming SMSs. Additionally, it has the capability to react to a few specific commands sent via SMS by an administrator (a member of the malicious gang).

As soon as the trojan is successfully installed on the phone, it sends an SMS message, 'App installed ok', to the default hard-coded administrator's phone number. In Zitmo, this was a UK phone number (+44778148xxxx[2]), which is now blocked by most operators. This SMS is sent only once, at the first startup after installation, probably to notify the

[2] Phone number censored.

attackers that they have a new victim. The phone number acts as an administrator number for the mobile trojan and pilots the rest of the attack.

Next, the malicious administrator needs to configure the trojan to target a particular online bank. To do so, he must send SMS messages to the victim with a specific body as text. The specific bodies of those SMS messages are also called commands.

So, the first command the administrator would have to send would ask the trojan to silently forward any SMS coming from the bank's phone number (add sender PHONENO). The second SMS would activate the spyware (ON).

When the victim logs into his bank account, or performs a sensitive online transaction, a malicious keylogger on the infected computer (Zbot binary on *Windows*) eavesdrops on the victim's credentials (e.g. login, password, credit card number). The first authentication factor falls into the hands of the Zeus gang. Meanwhile, the online bank sends an mTAN by SMS to the victim's phone. This SMS is intercepted by the mobile trojan, forwarded to the administrator's phone number and discarded from the victim's phone (step 4 of Figure 3). The second authentication factor falls into the hands of the Zeus gang.

Note that the mTAN is not displayed at all on the victim's phone, so he cannot correctly end his own banking operation. While the victim is probably still waiting for the mTAN, trying to figure out what is happening (or reverting to a degraded authentication mode without mTANs), the cybercriminals have all the information they need to conduct their fraud.

In fact, the real novelty in Zitmo is not the fact that it defeats two-factor authentication, but that it is able to initiate a fraudulent two-factor authentication. Indeed, prior to Zbot, executables were already capable of defeating mTANs: as mTANs are entered on a compromised PC, they merely needed to log keys or mouse gestures corresponding to the mTAN. What Zitmo improves is the fact cybercriminals can now choose to initiate the next frauds when they wish (they need no further interaction with the victim). For example, they can log into the victim's bank account (at any time), initiate a transfer, intercept the SMS from the victim's phone (at any time) and finish the transaction as if they were the user.

3.4 The making of Zitmo

Previously, we mentioned that Zitmo is very closely derived from a Russian spyware program named SMS Monitor. The code is very similar (several strings and assembly routines match exactly – see Table 1), with a few minor differences:

- Zitmo does not eavesdrop on outgoing SMS messages
- Zitmo sends an SMS the first time it is run
- Zitmo does not implement any licence control mechanism
- Zitmo's packaging is different: different name, different UID, and all features are grouped into a single executable (no DLLs).

The percentages in Table 1 clearly show that Zitmo is strongly related to SMS Monitor and SMS Monitor Lite. Note that there are also matches with other pieces of malware, such as SymbOS/Trapsms.A!tr.spy because, for instance, some object instantiation routines or string constants are very common among *Symbian* executables.

Like most software of this kind, it is advertised as a tool for 'parental control or security audit'. Although this is quite possibly a cover – a 'politically correct' way of describing a borderline application – the developer of SMS Monitor did not seem, to us, like a high-profile cybercriminal and/or someone related to the Zeus gang. From the outset, he was very helpful at responding to our inquiries, providing technical information about his spyware – which is not the behaviour one would expect of a criminal. Furthermore, it would not be very shrewd of him to sell a piece of spyware on the one hand (he says he has about 100 registered users) and incriminate himself by using it for malicious activities on the other.

It seems more likely that a member of the Zeus gang sought such a piece of spyware and either cracked a registered version or reverse engineered the lite version to re-use it in Zitmo. The task was probably made easier by the fact that the developer of SMS Monitor published two technical articles detailing his software in a Russian hacker magazine named 'xakep' (pronunciation close to 'hacker') [20, 21].

The Zeus gang re-packaged the spyware, naming the executable NokiaUpdate.exe so as not to arouse suspicion, and had it signed by *Symbian*'s *Express Signed* program [17]. The *Express Signed* program has already been abused by several pieces of malware (e.g. Yxes, CommDN, Album, NMPlugin) because it allows malware authors to sign their

Zitmo compared with	Exact match of assembly routines	Exact match of strings (case-sensitive match)
SMS Monitor Lite	60%	89%
SMS Monitor	59%	90%
SymbOS/Trapsms.A!tr.spy	13%	2%
SymbOS/Fwdsms.D!tr.spy	16%	30%

Table 1: Comparison of Zitmo with other trojans.

programs with all the security capabilities they should ever need, at an affordable cost [22]. Occasionally, a few applications submitted to the *Express Signed* program are selected for random audit but this was not the case for Zitmo, so the trojan was signed without anybody reviewing it.

To sign an application with *Express Signed*, one must, however, purchase a Publisher ID from *TrustCenter CA*. In the case of Zitmo, the malware authors provided *TrustCenter* with an official Azerbaijani company registration document and a copy of the alleged registrant's passport. Unfortunately, it seems that it is easy for members of the Zeus gang to get hold of fake or stolen identification – [21] even explains a possible procedure, scanning stolen documents.

In the second part of this article we will provide details of the reverse engineering of a *Symbian* Zitmo sample, as well as looking at security considerations and solutions.

ACKNOWLEDGEMENTS

We thank reviewers Guillaume Lovet (*Fortinet*) for his technical and in-depth review, and Ludovic Apvrille (*Telecom ParisTech*) for useful comments on the paper's structure. We also thank David Barroso (*s21sec*) for kindly sharing information regarding Zeus and Zitmo.

REFERENCES

[1] Hyppönen, M. Mobile phone threats. HITBSec Conference, 2005.

[2] de Haas, J. Symbian phone security. Black Hat Europe, 2005.

[3] Lovet, G. Dirty money on the wires: the business models of cybercriminals. Proceedings of the 16th Virus Bulletin Conference, 2006.

[4] Coursen, S. Mobile Malware: A Clear and Present Danger. ISSA, pp.35–36, January 2006. https://www.issa.org/Library/Journals/2006/January/Coursen%20-%20Mobile%20Malware.pdf.

[5] Chickowski, E. Chinese Mobile Malware Signals Danger Opportunity for Channel, April 2010. http://www.channelinsider.com/c/a/Security/Chinese-Mobile-Malware-Signals-Danger-Opportunity-for-Channel-590400/.

[6] Chen, T.; Peikari, C. Handbook of Research on Wireless Security, chapter 1. Idea Group Publishing. Y. Zhang, J. Zheng, M. Ma (eds.), 2008.

[7] Apvrille, A.; Zhang, J. Four Malware and a Funeral. 5th Conf. on Network Architectures and Information Systems Security (SAR-SSI), 2010.

[8] Barroso, D. ZeuS Mitmo: Man-in-the-mobile, September 2010. http://securityblog.s21sec.com/2010/09/zeus-mitmo-man-in-mobile-i.html.

[9] SymbOS/Zitmo.A!tr.spy. Fortiguard Center, Virus Encyclopedia, September 2010. http://www.fortiguard.com/encyclopedia/virus/symbos_zitmo.a!tr.spy.html.

[10] McDonald, D. Zeus: God of DIY Botnets, October 2009. http://www.fortiguard.com/analysis/zeusanalysis.html.

[11] Yang, K. ZeuS: Bot to Master early communication protocol (Part one of two), October 2010. http://blog.fortinet.com/zeus-bot-to-master-early-communication-protocol-12.

[12] Yang, K. ZeuS: Bot to Master early communication protocol (Part two of two), October 2010. http://blog.fortinet.com/zeus-bot-to-master-early-communication-protocol-part-two-of-two/.

[13] Apvrille, A. Symbian worm Yxes: Towards mobile botnets? 19th EICAR Annual Conference, 2010.

[14] Shub-Nigurrath. Primer in Reversing Symbian S60 Applications, June 2007. Version 1.4.

[15] Zhang, J. Find out the 'Bad guys' on the Symbian. Association of Anti Virus Asia Researchers Conference, 2007.

[16] Tarasov, D. SMS Monitor User Manual. http://dtarasov.ru/smsmonitor_manual_en.html.

[17] Apvrille, A. ZeuS In The MObile (Zitmo): Online Banking's Two Factor Authentication Defeated, September 2010. http://blog.fortinet.com/zeus-in-the-mobile-zitmo-online-bankings-two-factor-authentication-defeated/.

[18] Apvrille, A. Zitmo Follow Up: From Spyware to Malware, September 2010. http://blog.fortinet.com/zitmo-follow-up-from-spyware-to-malware/.

[19] Holz. T. Banking Malware 101. Chaos Communication Congress (25C3), December 2008.

[20] Tarasov, D. SMS Spy. xakep magazine, 9, 2006. http://www.xakep.ru/magazine/xa/093/132/1.asp (in Russian).

[21] Tarasov, D. Evil-coding Symbian. xakep magazine, 3, 2009. http://www.xakep.ru/magazine/xa/123/096/1.asp (in Russian).

[22] Apvrille, A. Symbian Signed Mobile Malware: One Gang?, July 2010. http://blog.fortinet.com/symbian-signed-mobile-malware-one-gang/.

FEATURE

CANADA'S NEW ANTI-SPAM LAW

John Levine
Taughannock Networks, USA

With the passage of bill C-28 in December, Canada became the last of the G-8 countries to make spamming illegal.

BACKGROUND OF C-28

For many years, Canada has had a privacy law called PIPEDA, which is similar to EU privacy laws. While this should have made most kinds of spam illegal (since it's illegal to trade in people's email addresses), in practice it had little effect – partly because its application to email was only implicit, and partly because the role of the Privacy Commissioner isn't designed to deal with uncooperative people. Ottawa professor Michael Geist successfully used a PIPEDA complaint to get a local sports team to stop sending him spam, but Montréal spam expert Neil Schwartzman has been unable to get a local spammer to stop sending advertisements for a worthless Canadian Subsidy Directory. (Neil tells me that the spammer dodged the Privacy Commissioner for at least a year by recognizing the Caller-ID and not answering the phone.)

In 2004, the government convened a Federal Anti-Spam Task Force (FAST-F). The task force filed its report in May 2005 [1], recommending among other things that Canada should pass a law that would clearly make spam illegal. The report received broad support, but for various political reasons it wasn't until April 2009 that a bill (C-27) was introduced.

C-27 incorporated most of the recommendations included in the 2005 report, and had made most of its way to passage – including the key committee hearings – when Parliament was prorogued in December 2009.

In May 2010, a new bill, C-28, was introduced [2]. This was almost identical to C-27 and moved easily through the committees until it was passed and signed into law on 15 December 2010.

The law is due to come into force around September 2011. As is typical in Canadian law, many details are left for the relevant ministry, Industry Canada, to write in regulations during the coming months. The law is quite long: 80 pages in the official bilingual version. Its length is largely due to a complex enforcement scheme which is spread among several existing agencies, as well as to carefully worded definitions and rules that attempt to draw a clear line between the allowed and the forbidden, and to some minor special case language added to placate various commercial constituencies. The authors of the bill were quite familiar with spam laws in other countries, and attempted to craft it to avoid the pitfalls that have been seen elsewhere.

Unlike CAN SPAM and other laws, C-28 has no snappy abbreviation, or any moniker shorter than its official title: 'An Act to promote the efficiency and adaptability of the Canadian economy by regulating certain activities that discourage reliance on electronic means of carrying out commercial activities, and to amend the Canadian Radio-television and Telecommunications Commission Act, the Competition Act, the Personal Information Protection and Electronic Documents Act and the Telecommunications Act'. Early versions had a short title which was deleted in committee (one of the few changes they made) so we'll just call it C-28.

THE CONSENT RULE

The basic rule of C-28 is that commercial email may only be sent to people who have given their consent to receive it. Needless to say, the definitions of 'commercial' and 'consent' are not brief.

The definition of a commercial electronic message includes, along with the expected email advertisements, any text or voice ads – the intent being to cover instant messaging and other online media that are not SMTP email. It also includes any request for permission to send commercial messages (so there's no 'may I send you this spam?' loophole) and the installation of software on another person's computer. It specifically excludes live telephone calls, faxes and voice mail, which are regulated separately under telemarketing laws.

The law states that a sender must have either explicit or implied consent to send mail. Explicit consent is what it sounds like – the recipient must have said it was OK to send them that kind of message. When requesting consent, the purpose for which consent is requested must be set out clearly, and it must be clear who is asking.

Consent to receive commercial email is implied under certain conditions: for example, an existing business or non-business relationship, or a message sent to someone who's published his address without a no-spam notice and the message is 'relevant to the person's business, role, functions or duties in a business or official capacity'. An existing business relationship is defined as having done business within two years, or sent an inquiry within six months.

'Business' is defined in some detail and includes barter and any kind of written contract. A 'non-business relationship' covers some special cases. It includes any political party, candidate or charity to whom one has made a donation, for whom one has done volunteer work, or attended a

meeting, within the past two years. It also includes any club or organization to which one has belonged in the past two years.

Messages must include a means by which to withdraw consent, i.e. unsubscribe, by the same route the message arrived (typically return email), as well as via a web link. The method must continue to work for at least 60 days after the message was sent, and the unsubscription must be effective within 10 business days.

Section 10 of the bill lays out the rules for obtaining express consent. The rules for downloads (which I helped develop) are quite long, due to the need to distinguish among various different types. Reasonable downloads, such as updates to software the user has already purchased, are allowed, and unreasonable ones, like toolbars hidden in packages with other software, are not. Any software can be installed with express consent. For downloads, the consent process must describe what the software does, what effect it will have on the computer, specifically including the collection of personal information, interfering with the owner's control of the computer, changing system settings, interfering with access to data on the computer, contacting other computers, and installing remote control software. Descriptions must be clear – no hiding the important parts in page 27 of an impenetrable boilerplate. Express consent is implied (yes, this is a bit tortured) if the download is a cookie, HTML code, JavaScript, an operating system, or a script executable by a previous download, so long as 'the person's conduct is such that it is reasonable to believe that they consent to the program's installation.'

Transitional rules apply for the first three years after the law comes into force, grandfathering previously existing business or non-business relationships. (Or, to put it another way, businesses have three years to reconfirm their mailing lists.)

OTHER ODDS AND ENDS

The act regulates some other online activities that don't have much in common with spam other than the fact that they're only acceptable if done with the user's consent.

For example, Section 7 forbids altering of 'the transmission data in an electronic message so that the message is delivered to a destination other than or in addition to that specified by the sender' without the express consent of the sender or recipient, or a court order, or for network management. This plugs a gap in wiretap law. The consent rules are the same as for mail, but with express consent only, no implied consent.

Section 8 regulates spyware and other downloads, discussed in more detail below.

Section 9 prohibits aiding, inducing, procuring, or causing to be procured anything that is forbidden in the earlier part of the bill. This short but important section follows the money, making it clear that the penalties for hiring a spammer are the same as for spamming directly. Other language makes company managers and officers responsible for the actions of their subordinates, thus removing the Casablanca defence ('I'm shocked, shocked') that has been effective in several CAN SPAM cases in the US.

Providers of downloads must specify an email address – good for at least a year – to which users can complain, and must assist users in removing the software without charge if the description was inaccurate.

C-28 amends the Competition Act to forbid false or misleading sender information or subject matter in an advertising electronic message, or a false or misleading representation in a 'locator', which generally means a URL. The Privacy Act is amended to outlaw address scraping software, the use of scraped addresses, and the collection of personal information via illegal remote access to a computer.

ENFORCEMENT

C-28 spreads enforcement among several existing government agencies. Most of it is enforced by the national telecom regulator, the Canadian Radio-television and Telecommunications Commission (CRTC), which is analogous to the Federal Communications Commission in the US or Ofcom in the UK.

The parts about false and misleading advertising are enforced by the Competition Commissioner, which is somewhat analogous to the Federal Trade Commission in the US, or the Office of Fair Trading in the UK. The parts related to privacy are enforced by the Privacy Commissioner, analogous to the Information Commissioner in the UK, and which (unfortunately) has no US analogue.

The act allows undertakings similar to consent agreements, in which someone admits to one or more violations, promises to stop (and perhaps pays a fine), and in return is given immunity to all further action for whatever prior actions he admitted to.

For less cooperative violators, the law gives the CRTC extensive powers to investigate violations, including orders to preserve records, and to conduct hearings that are in effect civil trials. Court injunctions are available, with serious penalties for failure to comply: up to $25,000 for individuals and $250,000 for organizations. The overall fines for one offence – which may include a lot of related

violations – can be up to $1 million for individuals and $10 million for companies.

Proceedings must start within three years of the violation, and a demonstration that the violator exercised due diligence to try to obey the law is specifically made a defence.

Unlike in most other countries, individuals who have received spam or been the victim of illegal software download have a private right of action (PROA) to sue violators. The same rules apply as in CRTC actions, such as the three-year time limit, undertaking immunity, and due diligence as a defence.

The court can award $200 per spam, up to $1 million per day. The Competition Act provisions have serious consequences, with penalties of up to $200,000 and years in prison for violations.

DISCLOSURE

Since Canada has a strong privacy law in PIPEDA, another lengthy section of the act describes the rules for disclosing and exchanging information relating to violations and investigations. Anyone can disclose information related to a violation to the CRTC, Competition or Privacy commissioners. The commissioners can disclose information related to an investigation to each other.

They are all allowed to disclose and exchange information with foreign agencies and international organizations (such as Interpol), so long as suitable protections are in place. The information can be related to Canadian cases, or to foreign cases under similar laws.

These provisions are very important, since most significant spam and malware activities are spread across multiple countries. In the past, Canadian investigators had been at a disadvantage due to a lack of certainty about what they were allowed to discuss with their foreign peers.

WHAT IT'S LIKELY TO MEAN

Depending on who you ask, C-28 is either the death of online advertising (and, by implication, the end of human civilization), or no big deal. The reality is somewhere in between.

Clearly, anyone who carries out commercial mailing to or from Canada will have to clean up their lists, and reconfirm them some time before 2014 if they don't have proper documentation that the people on the lists want to be on them. They'll also have to make sure that their confirmation and opt-out processes actually work. (You might think this would be obvious, but we've all seen plenty of evidence to the contrary.)

Anyone who provides downloadable software will similarly have to ensure that their installation time consent process is adequate, that people can remove the software, and that they have a process to receive and handle consumer complaints.

Canada will also be able to cooperate more with international anti-spam efforts – something that many Canadian law enforcement officials are eager to do. Cooperation is still somewhat limited pending the passage of companion bill C-29 [3] which would amend PIPEDA to provide more general privacy exceptions for law enforcement.

One of the biggest open questions is how much effect this will have on mailers and download vendors in the United States. Although Canadian law clearly does not apply in the US, the economies of the two countries are so intertwined that all but the smallest US companies do business in Canada. Their internets are equally intertwined – for example, although I'm in the US, I use a mail hosting company in Toronto for both US and Canadian clients, so all of their mail is subject to C-28.

Some observers think that C-28 makes the much weaker CAN SPAM act in the US obsolete or irrelevant, since US mailers will in practice all have to follow the stricter Canadian law. I'm not sure I'm ready to write off CAN SPAM yet, if for no other reason than that Americans are remarkably ignorant of other countries, even the one next door. There are certainly bulk mailers in the US who skate by with minimal CAN SPAM compliance who will find themselves in legal trouble when their mail inevitably ends up in Canada, and the recipients sue.

For the next six months, the main activity will be the writing of the regulations. After that, Canada will finally join the rest of the developed world and make spam illegal.

REFERENCES

[1] Stopping Spam: Creating a Stronger, Safer Internet Report of the Task Force on Spam. http://www.ic.gc.ca/eic/site/ecic-ceac.nsf/eng/h_gv00317.html.

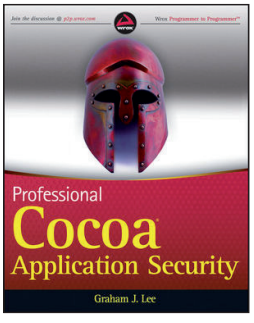
[2] C-28. http://www2.parl.gc.ca/Sites/LOP/LEGISINFO/index.asp?Language=E&Session=23&query=7019&List=toc.

[3] C-29 An Act to amend the Personal Information Protection and Electronic Documents Act (Safeguarding Canadians' Personal Information Act). http://www2.parl.gc.ca/Sites/LOP/LEGISINFO/index.asp?Language=E&query=7020&Session=23&List=toc.

BOOK REVIEW

A NICE DROP OF COCOA

David Harley
Mac Virus, UK



Title: Professional Cocoa Application Security

Author: Graham J. Lee

Publisher: Wrox (Wiley)

ISBN: 978-0-470-52595-1

Price: £33.99

Graham Lee is well known for his work on *Mac* and iGadgets (http://fuzzyaliens.com), his writing on security for the Mac Developer Network, and his blog at http://blog.securemacprogramming.com/. Recently, his book on application security was published by *Wrox* (a *Wiley* imprint) as part of a series of professional guides intended for programmers, developers and IT professionals.

The book isn't focused on malware: in fact, Lee hardly mentions malware at all except (briefly) in the introduction, and the only specific instance that he mentions is the formerly widespread, decidedly malicious, but reassuringly pre-*OS X* Autostart worm. Conceding that malware is much less of an issue on the *OS X* and *iOS* platforms (something of an understatement), he does not dismiss the importance of social engineering as a means of getting malcode installed onto a system, but that isn't a theme he develops in the rest of this book.

Rather, he stresses the developer's ability to enhance the customer's security by making use of *Apple*'s security tweaks in its implementation of Unix. His focus is on building security into the application right from the beginning of the development process (and not many of us are likely to argue with that).

COCOA AT COMPILE TIME, NOT BEDTIME

As well as being one of the comforting night-time beverages of choice for my generation, Cocoa is an object-oriented API for *Mac OS X* intended to automate the inclusion of functionality compliant with *Apple*'s human interface guidelines. As a result, the book is copiously dusted with examples of code in Objective-C, the weapon of choice for Cocoa development. (Example code is also available via the *Wrox* code download page, for those whose typing is apt to falter at compound terms like 'SCNetworkReachabilityCreateWithName' and barely-human-readable syntax.)

This is not a beginner's book – at any rate, it's not one for novice developers. While Lee doesn't assume that the reader is conversant with the basics of security or the development of secure applications, he does assume enough familiarity with C, Objective-C and the *Apple* APIs for the example code to make sense.

CHAPTER AND VERSE

Chapter one, 'Secure by Design', is (apart from a brief and abstract summary of the security implications of Cocoa) a platform-agnostic discussion of the principles of application security. It's a clear abstract description that could be useful to *any* newcomer to secure application development working on *any* OS. It covers the basics of risk profiling, security ergonomics and user psychology, asset audit, and threat classification and assessment applying the DREAD model (Damage, Reproducibility, Ease of attack, Affected users, Discoverability) and STRIDE threat classification (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege).

Chapter two, 'Managing Multiple Users', is pretty *OS X*-specific, since the Unix multi-user paradigm is effectively unused by the *iPhone* SDK. It does a good job of illustrating the similarities and differences between *OS X* and other Unix implementations, explaining the workings of Directory Services and the use of Kerberos for authentication.

Chapter three, 'Using the Filesystem Securely', is no substitute for *Apple*'s own, extensive historical documentation on the inner workings of the file system – but that isn't what a Cocoa developer needs. Instead, it summarizes the format and use of the Unix permission bits, setuid and setgid, HFS+ extensions, and Access Control Lists and the ACL API. It includes a high-level introduction to FileVault and other encryption options on *OS X* and on the *iPhone*, ranging from a high-level description of symmetric and asymmetric encryption to code illustrating a staged approach to building an encrypted file. After a brief description of network filesystems, there is a discussion of domains within the *OS X* filesystem, going on to security-specific issues such as the quarantine feature in Launch Services and secure deletion.

Chapter 4, 'Handling Multiple Processes', explains the need to design and implement multi-process systems in such a way as to reduce the risk of vulnerabilities being introduced during the development process – starting from the obvious area of privilege separation and touching on a related issue with the 'all or nothing' nature of the launchd daemon and the use of the setuid and setgid bits. The section on communication between processes reviews Mach ports and

the Distributed Objects IPC interface, and 'Playing in the Sandbox' looks at the public API to sandboxing in *OS X*. Again, *iPhone* sandboxing is not considered because access controls on iGadgets are mandatory, so there is no API for changing policy. The final section of this chapter looks at code signing services.

Chapter 5 looks at 'Storing Confidential Data in the Keychain', explaining what the keychain is and how to use it on both the *Mac* and the *iPhone*, explaining the differing functionality between the two platforms (*iPhone* keychain functionality is more limited) and the various uses for the facility (general passwords, Internet passwords, certificate management).

Chapter 6 returns to the subject of privilege, discussing the way in which an authorized application gains the right to perform a privileged task, and reducing the potential damage from a compromised application by dividing different privileges between separate processes. A description of the Authorization Database precedes a warning about the possible misuse of helper tasks as root, a summary of the padlock view, and finishes with a discussion of authorization plug-ins.

Chapter 7, 'Auditing Important Operations', goes beyond damage limitation to using auditing as a positive tool for reiteration of the threat-modelling process and for forensic use. It describes *OS X* facilities for audit using Authorization Services, firewall, sudo and user account logs, and change tracking. The author summarizes the advantages of using ASL (Apple System Logger) as the basis for an audit system. Finally, he briefly refers to NCSC's rainbow series of books and the more recent Common Criteria, though if conformity was important to a developer, he or she would need much more information than is available here before even considering how to implement a CC-conformant application.

Chapter 8 discusses 'Securing Network Connections' – an important topic in an era of remote applications and data. The first section looks at remote authentication using HTTP, and other authentication methods such as Kerberos, OpenID and OAuth. Several pages (including code samples) are devoted to service discovery and Bonjour, using NSNetService and NSNetServiceBrowser, with a high-level summary of the application of device pairing to the *iPhone*. The application firewall and ipfw packet filtering are described, but the text moves quickly on to network configuration with SystemConfiguration. SSL, however, is addressed at some length with a significant amount of example code.

Chapter 9, 'Writing Secure Application Code', focuses on defensive coding in Objective-C and C in general, and could be described as the meat of the book for a Cocoa developer – though some of the issues raised, such as problems with strcpy() and printf(), are far from specific to Objective-C. Nevertheless, this chapter offers a good start on considering the mitigation of insecure functions and such issues as safe buffer handling. The rest of the chapter looks at fairly formal aspects of defensive programming such as code review and code checklists and the use of static analysis tools, unit testing with Xcode, and automated testing with hardening tools: however, only one paragraph is devoted to fuzzing.

Chapter 10 is quite general, covering the secure deployment of software. This includes the provision of security documentation within the application and within the manual, the use of code signing to identify the provider, distribution and installation issues, the handling of bug reports, and software updating. Chapter 11 covers kernel extensions in just a few pages, and chapter 12 briefly summarizes the threat model that underpins the book and offers some suggestions for further reading.

CONCLUSION

A former manager of mine once cited the CIA tripod model of security (Confidentiality, Integrity, Availability) and said (understandably) that as far as he was concerned, availability trumped the other two. This seems to be a cornerstone of Lee's view of security too: while his quintessential summary of application security is 'Know your app and its users', his emphasis on making apps 'appropriately secure' is supplemented by his awareness – made explicit at several points in the text – that the most secure program in the world is useless if it's unusable, and that it doesn't matter how secure data is in other respects, if it can't be accessed by the people who are authorized to make use of it.

Though it's by no means a complete reference, this book is a good, clear read, combining general principles of application security and defensive programming with enough Cocoa-specific information and sample code to provide an *OS X* developer with a basis for enhancing the security of his applications. An *iPhone* developer would, I think, need to do rather more in the way of additional research. On the other hand, any application developer with an interest in defensive programming could do worse than take a look through this as primer material. It's easy (and rather common) to overstate the impact of attacks that exploit vulnerabilities in software rather than the naïveté of users. However, readers of this book can make a significant difference to the welfare of the *Apple*-using community if it inspires them to think more proactively about reducing risks to their customers by reducing the likelihood of inherent weaknesses in their applications that lead to common exploit types.

COMPARATIVE REVIEW

VBSPAM COMPARATIVE REVIEW MARCH 2011

Martijn Grooten

Those who are familiar with our VB100 tests, where the criteria to pass the test are to have a 100% catch rate combined with zero false positives, may wonder why we don't have similar criteria for the VBSpam tests. The reason goes deeper than the simple fact that no product has ever achieved this.

All of the tens of thousands of spam messages sent as part of this test were originally sent to fictional addresses that have never belonged to a real person or organization (spam traps). Messages sent to these addresses are, by definition, unwanted: there is no one to have wanted them in the first place.

However, that does not mean that a handful of messages in the spam corpus may not appear rather legitimate: for instance, this month's spam corpus contained a genuine invite to join *Facebook*, and a newsletter that looks as if it will be of genuine interest to some recipients.

So while the *right* thing to do with these messages would be to block them (as, ultimately, a spam filter should block what is unwanted), this may not be the *best* thing to do. Blocking *Facebook* invitations – the vast majority of which are legitimate and wanted – will likely lead to false positives. Blocking a newsletter, even if its sender did not adhere to best practices, may also lead to complaints from end-users.

We have always believed that avoiding false positives is more important than blocking as much spam as possible, and it is good to see there were no false positives for any of the products with the top seven final scores.

Still, we acknowledge that false positives may occur from time to time, and sometimes for a good reason. Some of the legitimate emails that were missed during this test contained URLs on domains frequently used by spammers. And while the best thing to do with such messages would be to allow them through to the user's inbox, marking them as spam is certainly understandable. As long as it does not occur frequently, this should not prevent a product from winning a VBSpam award.

In this test 18 out of 19 full solutions (hailing from 13 different countries – a nice record showing the global nature of the fight against spam) achieved VBSpam certification, eight of which had no false positives.

THE TEST SET-UP

The VBSpam test methodology can be found at http://www.virusbtn.com/vbspam/methodology/. As usual, email was sent to the products in parallel and in real time, and products were given the option to block email pre-DATA. Four products chose to make use of this option.

As in previous tests, the products that needed to be installed on a server were installed on a *Dell PowerEdge* R200, with a 3.0GHz dual core processor and 4GB of RAM. The *Linux* products ran on *SuSE Linux Enterprise Server 11*; the *Windows Server* products ran on either the 2003 or the 2008 version, depending on which was recommended by the vendor.

To compare the products, we calculate a 'final score', which is defined as the spam catch (SC) rate minus five times

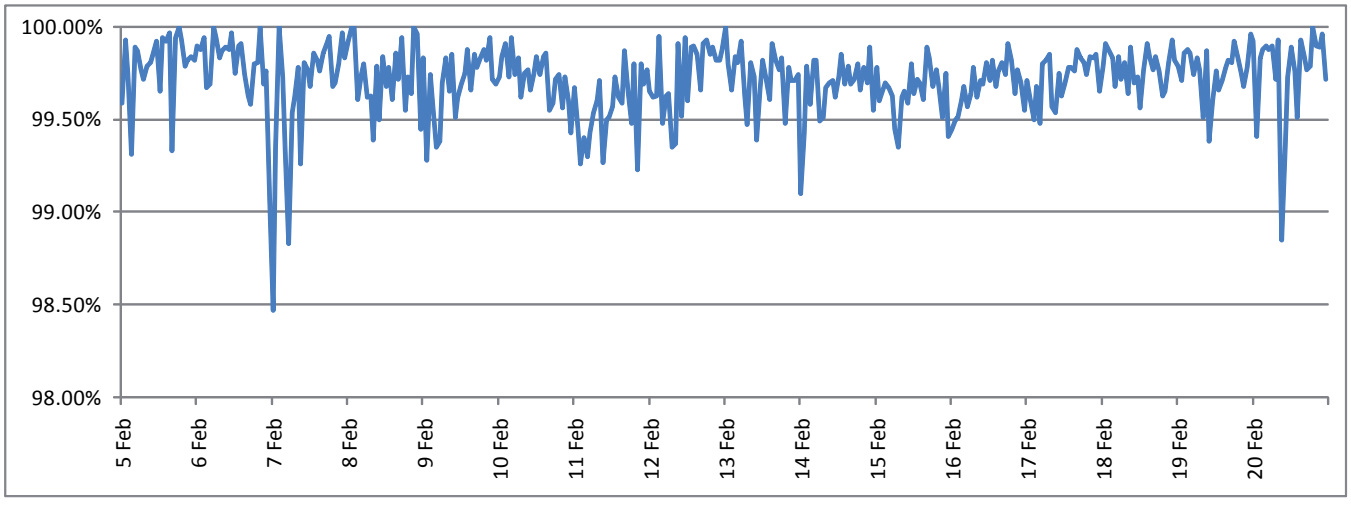


Figure 1: Average catch rate of all full solutions throughout the test.

the false positive (FP) rate. Products earn VBSpam certification if this value is at least 97:

SC - (5 x FP) ≥ 97

THE EMAIL CORPUS

The test ran for 16 consecutive days, from around 12am GMT on Saturday 5 February 2011 until 12am GMT on Monday 21 February 2011.

The corpus contained 140,800 emails, 137,899 of which were spam. Of these, 80,755 were provided by *Project Honey Pot* and 57,144 were provided by *Abusix*; in both cases, the messages were relayed in real time, as were the 2,901 legitimate emails – a record number for our tests. As before, the legitimate emails were sent in a number of languages to represent an international mail stream.

Figure 1 shows the average catch rate of all full solutions throughout the test. As before, to avoid the average being skewed by a poorly performing product, we excluded the highest and lowest catch rate for each hour.

In the previous test we noticed the somewhat surprising fact that spam sent in plain text appeared to be significantly more difficult to filter than spam with both HTML and text in the body or with a pure HTML body. This month's results showed that this was still the case – though, again, we stress that this does not necessarily mean that the messages are difficult to filter *because* of the plain text body. Still, developers looking into ways to improve their products' performance may want to have a look at whether the filtering of text-only emails can be improved upon.

In this test, we again looked at the overall spam corpus and compared it with the sub-corpus of 'difficult' spam – defined as those messages missed by at least two different filters; the latter concerned slightly more than 1 in 42 messages.

This time we looked at the size of the emails (header plus body; in bytes) and how the distribution of sizes differed between the two corpora. Ordering the emails in both corpora by size, Figure 2 shows how quickly emails get bigger in each corpus.

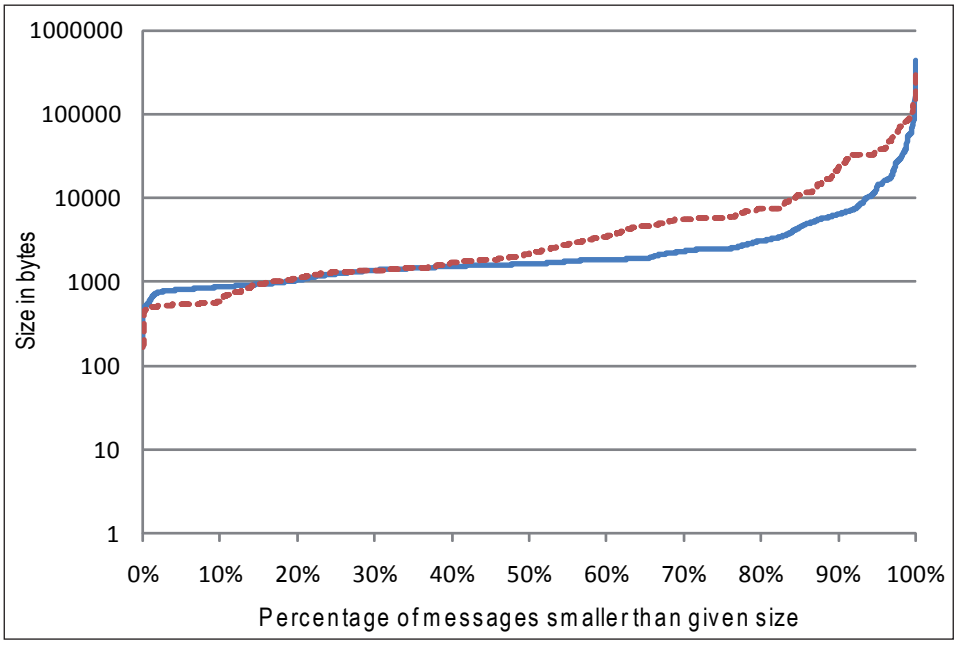


Figure 2: Size of spam emails.

The blue line corresponds to the full corpus, the red one to that of more difficult spam. The graph, with the bytes shown on the vertical axis in logarithmic scale should be read as follows: (exactly) 80% of all spam (blue line) was less than 3,072 bytes in size, while 80% of the difficult spam had a size of less than 7,222 bytes. In the latter corpus, just 57% was smaller than 3,072 bytes.

What the graph shows is that both very small and very large spam messages are harder to filter. It would be wrong to conclude that this difficulty in filtering is a consequence of the message size, but it is nevertheless something developers may want to keep in mind when trying to improve their products.

RESULTS

AnubisNetworks Mail Protection Service

SC rate: 99.83%
FP rate: 0.00%
Final score: 99.83
Project Honey Pot SC rate: 99.84%
Abusix SC rate: 99.82%

The increase in the global volume of spam that was seen just after the beginning of the year would not have been felt by the customers of

AnubisNetworks, as the product's spam catch rate improved significantly too. Thankfully, this improvement was without any false positives, and with the third highest final score the product wins its fifth consecutive VBSpam award.

BitDefender Security for Mail Servers 3.0.2

SC rate: 99.78%
FP rate: 0.00%
Final score: 99.78
Project Honey Pot SC rate: 99.78%
Abusix SC rate: 99.78%

A participant since the very first test, *BitDefender* continues to be the only product to have won a VBSpam award in every test. Despite the product's impressive track record, its developers are not ones for resting on their laurels, as was demonstrated by an improvement in the product's final score for the third time in a row. With no false positives, the product achieved the fifth highest final score and another well deserved VBSpam award.

eleven eXpurgate Managed Service 3.2

SC rate: 99.62%
FP rate: 0.00%
Final score: 99.62
Project Honey Pot SC rate: 99.43%
Abusix SC rate: 99.89%

I was pleased to see *eleven*'s *eXpurgate* return to the test, having been absent since July 2010. Its developers explained that, for them, reducing false positives is an absolute priority and they try to achieve this, among other things, by classifying email into several categories: ham, spam, bulk, almost empty etc. Our test only distinguishes between ham and spam – as do many end-users – but this was not a problem for the product. It did not miss a single legitimate email and it won a VBSpam award with a very decent and improved final score.

Fortinet FortiMail

SC rate: 99.82%
FP rate: 0.00%
Final score: 99.82
Project Honey Pot SC rate: 99.81%
Abusix SC rate: 99.85%

It is a good thing if a product performs well several tests in a row, and even better when it also manages to improve its performance. In this test, *FortiMail* both increased its spam catch rate slightly and eliminated false positives. The appliance saw its final score improve for the third time in a row and easily wins its 11th VBSpam award.

Halon Mail Security

SC rate: 99.71%
FP rate: 0.00%
Final score: 99.71
Project Honey Pot SC rate: 99.72%
Abusix SC rate: 99.69%

Halon, a Swedish company, offers three spam-filtering solutions: a hosted solution, a hardware appliance and a virtual appliance; we tested the latter.

The product was installed easily on *VMware* and setting it up was equally straightforward. A nice, intuitive web interface lets the administrator change settings where necessary and I was particularly charmed by the fact that the product uses its own scripting language that can be used to fine-tune its performance. This will give more tech-savvy sysadmins the possibility to add their own rules to tailor the product specifically for their organization's particular needs.

Of course, a user-friendly interface is only meaningful if the product itself performs well, but that was certainly the case here. With a good spam catch rate and no false positives at all, *Halon Mail Security*'s final score is among the highest in this test and earns the product a well-deserved VBSpam award.

Kaspersky Anti-Spam 3.0

SC rate: 98.32%
FP rate: 0.00%
Final score: 98.32
Project Honey Pot SC rate: 98.75%
Abusix SC rate: 97.73%

Kaspersky's spam catch rate was significantly lower this month than in the previous test thanks to a run of bad days during the test, when the product's performance dropped quite a bit (as can

be seen from the high standard deviation in the table on p.21). Thankfully, there was room for this, and as there were no false positives, the product ended up winning its tenth VBSpam award.

Libra Esva 2.0

SC rate: 99.89%
SC rate pre-DATA: 98.34%
FP rate: 0.00%
Final score: 99.89
Project Honey Pot SC rate: 99.84%
Abusix SC rate: 99.95%

Libra Esva achieved the second highest final score in the previous test – and managed to repeat the achievement this month, once again combining a very high spam catch rate with a zero false positive rate. The Italian company earns its sixth VBSpam award in as many tests.

McAfee Email Gateway (formerly IronMail)

SC rate: 99.92%
FP rate: 0.21%
Final score: 98.88
Project Honey Pot SC rate: 99.87%
Abusix SC rate: 99.99%

As in the previous test, *McAfee*'s *Email Gateway* appliance achieved the second highest spam catch rate. Unfortunately, some of the domains that were seen in legitimate emails but which are also frequently used in spam messages, caused the product to generate a handful of false positives. Still, with a decent final score, the product earns its tenth consecutive VBSpam award.

McAfee Email and Web Security Appliance

SC rate: 99.20%
FP rate: 0.48%
Final score: 96.79
Project Honey Pot SC rate: 99.16%
Abusix SC rate: 99.27%

With 14 false positives, *McAfee*'s second appliance had the joint highest false positive rate. This would not automatically have denied the product a VBSpam award, but with a spam catch rate that was slightly below average, the product's final score fell below the threshold of 97 and no certification can be awarded.

MessageStream

SC rate: 99.70%
FP rate: 0.48%
Final score: 97.29
Project Honey Pot SC rate: 99.63%
Abusix SC rate: 99.81%

MessageStream's hosted solution missed a relatively high number of legitimate emails, but its spam catch rate was high enough to make up for that. There is certainly room for improvement, but in the meantime, *MessageStream* wins its 11th VBSpam award.

OnlyMyEmail's Corporate MX-Defender

SC rate: 99.99%
FP rate: 0.14%
Final score: 99.30
Project Honey Pot SC rate: 99.99%
Abusix SC rate: 99.98%

OnlyMyEmail's *MX-Defender* continues to amaze me with its spam catch rate, which this time saw it miss less than one in 8,500 spam emails. Unlike in the previous test, there were a few false positives this time, which reduced the final score, but it was still a very decent score and wins the product its third VBSpam award.

Sophos Email Appliance

SC rate: 99.91%
FP rate: 0.00%
Final score: 99.91
Project Honey Pot SC rate: 99.87%
Abusix SC rate: 99.97%

Sophos's previous test result contains a lesson for all users of spam filters. The product was set up to tag spam emails with an 'X-Spam: yes' header. What looked like and counted as its single false positive in the previous test was actually an email that already contained this header – probably added by an

	True negatives	False positives	FP rate	False negatives	True positives	SC rate	Final score
Anubis Networks	2901	0	0.00%	230	137669	99.83%	99.83
BitDefender	2901	0	0.00%	305	137594	99.78%	99.78
eleven	2901	0	0.00%	521	137378	99.62%	99.62
FortiMail	2901	0	0.00%	244	137655	99.82%	99.82
Halon Mail Security	2901	0	0.00%	400	137499	99.71%	99.71
Kaspersky Anti-Spam	2901	0	0.00%	2311	135588	98.32%	98.32
Libra Esva	2901	0	0.00%	155	137744	99.89%	99.89
McAfee Email Gateway	2895	6	0.21%	113	137786	99.92%	98.88
McAfee EWS	2887	14	0.48%	1097	136802	99.20%	96.79
MessageStream	2887	14	0.48%	408	137491	99.70%	97.29
OnlyMyEmail	2897	4	0.14%	16	137883	99.99%	99.30
Sophos Email Appliance	2901	0	0.00%	122	137777	99.91%	99.91
SPAMfighter	2895	6	0.21%	2179	135720	98.42%	97.39
SpamTitan	2894	7	0.24%	143	137756	99.90%	98.69
Symantec Brightmail Gateway	2899	2	0.07%	183	137716	99.87%	99.52
The Email Laundry	2895	6	0.21%	413	137486	99.70%	98.67
Vade Retro	2896	5	0.17%	240	137659	99.83%	98.96
Vamsoft ORF	2901	0	0.00%	1245	136654	99.10%	99.10
Webroot	2893	8	0.28%	185	137714	99.87%	98.49
Spamhaus ZEN+DBL*	2901	0	0.00%	1768	136131	98.72%	98.72

**As the only partial solution in this test, the results for Spamhaus are listed separately from the full solutions.*

over-zealous outbound filter. While for products in our tests it shows the importance of adding more unique headers, all users of spam filters should know that even a very good spam filter can perform suboptimally because of a minor tweak in its settings.

And *Sophos Email Appliance* is a very good spam filter, as demonstrated in the current test. The third highest spam catch rate combined with no false positives at all gave it the highest final score and the product's developers in the UK and Canada should consider themselves the winners of this test.

SPAMfighter Mail Gateway

SC rate: 98.42%
FP rate: 0.21%
Final score: 97.39
Project Honey Pot SC rate: 99.06%
Abusix SC rate: 97.52%

SPAMfighter's developers will be keen to learn the reason for their product's reduced spam catch rate – which they no doubt will be a little disappointed with. This was mostly due to a large number of missed spam in the *Abusix* corpus. Ultimately, they will be pleased to learn that the spam catch rate was still high enough that even a handful of false positives did not get in the way of them winning their ninth VBSpam award.

SpamTitan

SC rate: 99.90%
FP rate: 0.24%
Final score: 98.69
Project Honey Pot SC rate: 99.94%
Abusix SC rate: 99.83%

SpamTitan continues to score one of the highest spam catch rates in the tests, and users of the virtual appliance will

	Project Honey Pot		Abusix		pre-DATA†		STDev‡
	FN	SC Rate	FN	SC Rate	FN	SC Rate	
Anubis Networks	128	99.84%	102	99.82%	N/A	N/A	0.54
BitDefender	179	99.78%	126	99.78%	N/A	N/A	0.58
eleven	460	99.43%	61	99.89%	N/A	N/A	1.18
FortiMail	156	99.81%	88	99.85%	N/A	N/A	0.47
Halon Mail Security	224	99.72%	176	99.69%	N/A	N/A	0.39
Kaspersky Anti-Spam	1013	98.75%	1298	97.73%	N/A	N/A	4.79
Libra Esva	128	99.84%	27	99.95%	2285	98.34%	0.23
McAfee Email Gateway	105	99.87%	8	99.99%	N/A	N/A	0.20
McAfee EWS	681	99.16%	416	99.27%	N/A	N/A	1.07
MessageStream	302	99.63%	106	99.81%	N/A	N/A	0.45
OnlyMyEmail	6	99.99%	10	99.98%	N/A	N/A	0.05
Sophos Email Appliance	104	99.87%	18	99.97%	N/A	N/A	0.24
SPAMfighter	759	99.06%	1420	97.52%	N/A	N/A	3.01
SpamTitan	48	99.94%	95	99.83%	N/A	N/A	0.19
Symantec Brightmail Gateway	130	99.84%	53	99.91%	N/A	N/A	0.33
The Email Laundry	332	99.59%	81	99.86%	1177	99.15%	0.45
Vade Retro	195	99.76%	45	99.92%	N/A	N/A	0.42
Vamsoft ORF	914	98.87%	331	99.42%	N/A	N/A	0.72
Webroot	84	99.90%	101	99.82%	32948	76.11%	0.22
Spamhaus ZEN+DBL*	1113	98.62%	655	98.85%	2805	97.97%	0.94

** As the only partial solution in this test, the results for Spamhaus are listed separately from the full solutions.*

† pre-DATA filtering was optional and was applied on the full spam corpus. All of The Email Laundry's false positives occurred pre-DATA; none of the other products had pre-DATA false positives.

‡ The standard deviation of a product is calculated using the set of its hourly spam catch rates.

find few spam emails in their inboxes. In this test, the high SC rate came at the expense of seven blocked legitimate emails. While these are seven too many, they were not enough to prevent the product from winning yet another VBSpam award.

Symantec Brightmail Gateway 9.0

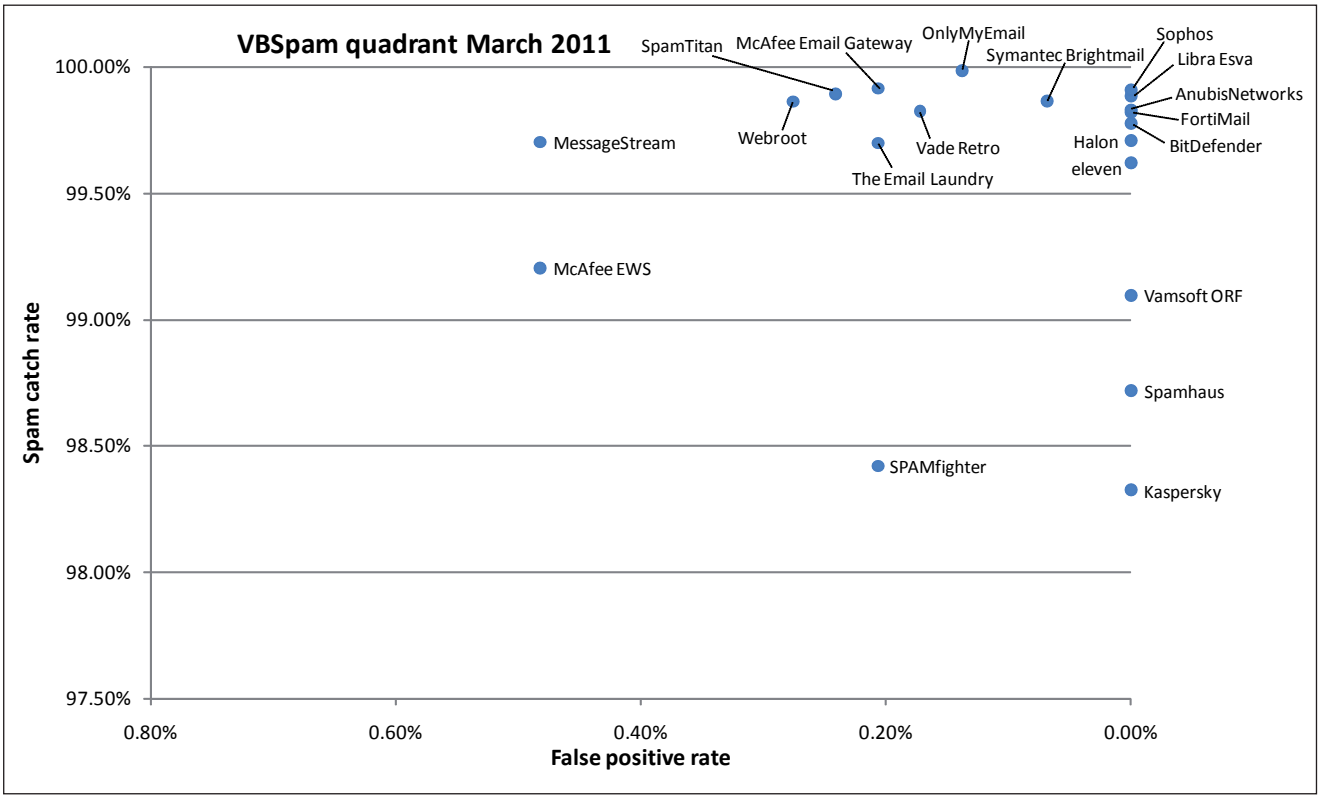
SC rate: 99.87%
FP rate: 0.07%
Final score: 99.52
Project Honey Pot SC rate: 99.84%
Abusix SC rate: 99.91%

Two related legitimate emails were missed on the first day of the test and these got in the way of *Symantec*'s virtual appliance achieving an even better final score. The product's developers can console themselves with the fact that their final score was the highest among those products that had false positives, and they can add yet another VBSpam award to their unbroken series.

The Email Laundry

SC rate: 99.70%
SC rate pre-DATA: 99.15%
FP rate: 0.21%
Final score: 98.67
Project Honey Pot SC rate: 99.59%
Abusix SC rate: 99.86%

One of the rules in this test is that we count no more than four false positives



per sending IP address. We believe that, in practice, multiple blocked messages from the same legitimate sender will either result in the sender finding a different way to communicate their message, or the recipient will adjust their filter – for instance by whitelisting the sender's address.

This explains why *The Email Laundry*, which missed a few dozen legitimate emails from one sender, only scored six false positives in this test and thus easily won its sixth VBSpam award. More important was the fact that, well before the end of the test, and before we had had a chance to give the developers feedback on the product's performance, emails from the sender in question were being accepted.

Vade Retro Center

SC rate: 99.83%
FP rate: 0.17%
Final score: 98.96
Project Honey Pot SC rate: 99.76%
Abusix SC rate: 99.92%

Vade Retro Center scored its highest spam catch rate to date, and while it was sad to see that the number of false positives increased to five, a sixth consecutive VBSpam award was easily won by the product.

Vamsoft ORF

SC rate: 99.10%
FP rate: 0.00%
Final score: 99.10
Project Honey Pot SC rate: 98.87%
Abusix SC rate: 99.42%

For the fourth time in six tests, *ORF* did not miss a single legitimate email – a unique achievement among full solutions in this test. Even if the spam catch rate isn't quite as high as that of some other products, the customers of the Hungarian-developed product will have little reason to look in their spam folders, and that may be just as important. A sixth VBSpam award is thus well deserved.

Webroot

SC rate: 99.87%
SC rate pre-DATA: 76.11%

Products ranked by final score	Final score
Sophos Email Appliance	99.91
Libra Esva	99.89
Anubis Networks	99.83
FortiMail	99.82
BitDefender	99.78
Halon Mail Security	99.71
eleven	99.62
Symantec Brightmail Gateway	99.52
OnlyMyEmail	99.30
Vamsoft ORF	99.10
Vade Retro	98.96
McAfee Email Gateway	98.88
Spamhaus ZEN+DBL	98.72
SpamTitan	98.69
The Email Laundry	98.67
Webroot	98.49
Kaspersky Anti-Spam	98.32
SPAMfighter	97.39
MessageStream	97.29
McAfee EWS	96.79

Webroot contd.

FP rate: 0.28%
Final score: 98.49
Project Honey Pot SC rate: 99.90%
Abusix SC rate: 99.82%

Once again, *Webroot*'s hosted anti-spam solution blocked the vast majority of spam emails, significantly reducing its customers' need for bandwidth. Unfortunately, there were a number of false positives which meant the product achieved a slightly lower final score, but it still performed well enough to earn its 11th consecutive VBSpam award.

Spamhaus ZEN+DBL

SC rate: 98.72%
SC rate pre-DATA: 97.97%
FP rate: 0.00%
Final score: 98.72
Project Honey Pot SC rate: 98.62%
Abusix SC rate: 98.85%

It is hard to think of spam filtering without the use of DNS blacklists and, thinking about those, it is hard to ignore *Spamhaus*. Another good performance showed that there is a reason for this: the *ZEN* IP-based blacklist blocked just short of 98% of all spam, while subsequent scanning of email bodies against the *DBL* domain blacklist blocked over one third of the remaining emails. Both were without false positives and the product – which is only a partial solution – won its eighth VBSpam award in as many tests.

CONCLUSION

It was pleasing to award VBSpam certification to 19 products this month and to find out that the stricter threshold we introduced in the previous test does not pose too many difficulties for the products. A number of new products are expected to join the field for the next test and they will have their work cut out if they are to match the standards set by the current entrants.

In the introduction to this review, I mentioned the difference between the 'right' decision and the 'best' decision for a spam filter to make on a particular email, and how these two usually, but not always, coincide. To overcome this problem, users and system administrators might want to whitelist or blacklist certain senders, IP addresses and domains. In future tests, we hope to be able to verify whether products have this option available and whether it works.

Performance tables from this test, and each of the 11 previous tests can be viewed on the redesigned VBSpam website at http://www.virusbtn.com/vbspam.

Later in March I will be discussing the subject of testing messaging filters in a slightly broader context at the eCrime Researchers Sync-Up, organized by the Anti-Phishing Working Group. Details of the event, which will be held 14–15 March in Dublin, can be found at http://www.ecrimeresearch.org/2011syncup/agenda.html.

The next VBSpam test will run in April. Developers interested in submitting their products should contact me on martijn.grooten@virusbtn.com.

END NOTES & NEWS

The 12th annual CanSecWest conference will be held 9–11 March 2011 in Vancouver, Canada. See http://cansecwest.com/.

The 8th Annual Enterprise Security Conference will be held 14–15 March 2011 in Kuala Lumpur, Malaysia. For details see http://www.acnergy.com/.

Black Hat Europe takes place 15–18 March 2011 in Barcelona, Spain. For more information see http://www.blackhat.com/.

Infosecurity Europe will take place 19–21 April 2011 in London, UK. For more details see http://www.infosec.co.uk/.

SOURCE Boston 2011 will be held 20–22 April 2011 in Boston, MA, USA. For more details see http://www.sourceconference.com/.

The New York Computer Forensics Show will be held 26–27 April 2011 in New York, NY, USA. For more information see http://www.computerforensicshow.com/.

The 5th International CARO Workshop will be held 5–6 May 2011 in Prague, Czech Republic. The main theme of the conference will be 'Hardening the net'. Details are available on the conference website at http://www.caro2011.org/.

The 20th Annual EICAR Conference will be held 9–10 May 2011 in Krems, Austria. This year's conference is named 'New trends in malware and anti-malware techniques: myths, reality and context'. A pre-conference programme will run 7–8 May. For full details see http://www.eicar.org/conference/.

The 6th International Conference on IT Security Incident Management & IT Forensics will be held 10–12 May 2011 in Stuttgart, Germany. See http://www.imf-conference.org/.

TakeDownCon takes place 14–19 May 2011 in Dallas, TX, USA. The event aims to bring together security researchers from corporate, government and academic sectors as well the underground to present and debate the latest security threats and disclose and scrutinize vulnerabilities. For more details see http://www.takedowncon.com/.

The 2nd VB 'Securing Your Organization in the Age of Cybercrime' Seminar takes place 24 May 2011 in Milton Keynes, UK. Held in association with the MCT Faculty of The Open University, the seminar gives IT professionals an opportunity to learn from and interact with security experts at the top of their field and take away invaluable advice and information on the latest threats, strategies and solutions for protecting their organizations. For details see http://www.virusbtn.com/seminar/.

The 2011 National Information Security Conference will be held 8–10 June 2011 in St Andrews, Scotland. Registration for the event is by qualification only – applications can be made at http://www.nisc.org.uk/.

The 23rd Annual FIRST Conference takes place 12–17 June 2011 in Vienna, Austria. The conference promotes worldwide coordination and cooperation among Computer Security Incident Response Teams. For more details see see http://conference.first.org/.

SOURCE Seattle 2011 will be held 16–17 June 2011 in Seattle, WA, USA. For more details see http://www.sourceconference.com/.

Black Hat USA takes place 30 July to 4 August 2011 in Las Vegas, NV, USA. DEFCON 19 follows the Black Hat event, taking place 4–7 August, also in Las Vegas. For more information see http://www.blackhat.com/ and http://www.defcon.org/.

The 20th USENIX Security Symposium will be held 10–12 August 2011 in San Francisco, CA, USA. See http://usenix.org/.

VB2011 takes place 5–7 October 2011 in Barcelona, Spain. *VB* is currently seeking submissions from those wishing to present at the conference. Full details of the call for papers are available at http://www.virusbtn.com/conference/vb2011. For details of sponsorship opportunities and any other queries relating to VB2011, please contact conference@virusbtn.com.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues):

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

Corporate rates include a licence for intranet publication.

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

Editorial enquiries, subscription enquiries, orders and payments:
Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire OX14 3YP, England
Tel: +44 (0)1235 555139 Fax: +44 (0)1865 543153
Email: editorial@virusbtn.com Web: http://www.virusbtn.com/