NOVEMBER 2010

virus BULLETIN

Fighting malware and spam

CONTENTS

2 **COMMENT**
Worldwide anti-botnet initiatives

3 **NEWS**
Wot no comparative?
Anti-botnet operations

3 **VIRUS PREVALENCE TABLE**

4 **CONFERENCE REPORT**
Vancouver expedition

8 **MALWARE ANALYSIS**
Deelaed learning

11 **TUTORIAL**
Exploit identification

14 **TECHNICAL FEATURE**
Anti-unpacker tricks – part fourteen

18 **LETTERS**

20 **END NOTES & NEWS**

ISSN 1749-7027

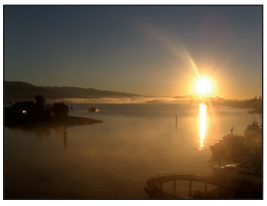
IN THIS ISSUE

IF YOU DON'T CARE, WE'LL MAKE YOU CARE

Sorin Mustaca describes some worldwide anti-botnet initiatives and urges ISPs, governments, ISVs and security companies to work together to control the plague of the Internet world.

page 2

A ~~STUXNET~~ BIRTHDAY PARTY TO REMEMBER



The 20th VB conference drew to a close last month in the beautiful city of Vancouver. Helen Martin reports on the presentations, games, awards, birthday cakes, and the buzzword of the event: Stuxnet.

page 4

FLASH IN THE PAN?

Over a period of around three months, new virus writer hh86 produced a handful of viruses using some new techniques – then disappeared without a word. Peter Ferrie details her creations.

page 8

vb

'Germany was the first country to launch a large-scale malware-cleaning project backed by the government.'

Sorin Mustaca, Avira

WORLDWIDE ANTI-BOTNET INITIATIVES

Millions of computers around the world are being controlled by malicious programs unbeknownst to their owners. These computers are used to send spam and phishing emails, spread malware, steal information and launch DDoS attacks on various networks. Take many computers infected by the same malicious program and controlled by the same people and you have a botnet.

At the beginning of October, Scott Charney, Corporate Vice President of *Microsoft*'s Trustworthy Computing team, suggested that infected machines should be blocked from the Internet and kept in quarantine until they are given clearance from some authority (ISP or governmental entity). Setting aside the obvious arguments as to how ISPs and governments would identify and isolate the infected PCs, I feel that Charney's idea unfairly punishes the victim – the owner of the PC – for being infected. His proposal suggests that the victim is to blame, rather than faulty or badly written software (e.g. the operating system, third-party programs). Anyone who has been in the computer security industry for long enough will know that the truth is somewhere in the middle. The owners of the affected machines are not infecting their computers deliberately. They get infected because they are connected – through an ISP – to the Internet. So, does this mean that we can identify those who are responsible for this mess? Or those who can do something to help solve it? The ISPs?

I can't answer these questions, but there are some who have started to take action. In the last three years we have seen a number of governmental initiatives that tried to opt in big ISPs with the purpose of identifying and cleaning infected PCs.

Germany was the first country to launch a large-scale malware-cleaning project backed by the government, ISPs and security companies. The 'Botfrei' ('Bot Free') initiative is a cooperation between eco (Association of the German Internet Industry) and the German Federal Bureau for Information Security (BSI). There is a telephone hotline for anyone who thinks their computer may be infected. The major ISPs in Germany are also cooperating (*1und1*, *Telekom*, *Kabel BW*, *NetCologne*, *QSC* and *Versatel*). They monitor suspicious activity on all IP addresses in their pool. For example, the sending of large volumes of data on port 25 for SMTP and incoming HTTP connections are considered suspicious activities. Once an ISP detects such activity, the customer is sent an email notification describing the suspicious activity and providing various other details. Anti-virus solutions are recommended to those who are infected.

Germany is not alone in taking such steps. One of the UK's biggest ISPs, *Virgin Media*, has started to notify customers whose machines appear to be part of a botnet. The customers receive instructions on how to install free AV solutions, how to clean their computers and how to keep them clean. If unable to do this themselves, the customers can opt for a paid service provided by the ISP.

Comcast, the largest residential cable and Internet service provider in the United States, started a botnet-detection service a year ago and is planning to extend this to over 16 million customers in the next few months. The service detects connections made to botnet command centres and displays a banner on the user's browser if they make connections there. Moreover, the Federal Communications Commission (FCC) is working on creating a 'Cybersecurity Roadmap' aiming to identify weaknesses in the country's Internet infrastructure and to identify threats to home, corporate and government networks.

Another pioneer in this field is the Australian Internet Industry Association (IIA) which has already drafted an 'eSecurity Code' to which ISPs can adhere voluntarily.

Even though this kind of initiative has received a lot of positive press and support, it is only the beginning, so results will be slow to appear. Nevertheless, I raise my hat to these initiatives and hope that ISPs, governments, ISVs and software security companies will continue to work together to control this plague of the Internet world.

NEWS

WOT NO COMPARATIVE?

Regular *VB* readers will spot that something is missing from this month's issue. Unfortunately, several network issues have introduced significant delays to this month's VBSpam test. In the interest of avoiding further delays to the publication of this issue of *VB*, the results of the VBSpam test will now be published in the December issue (alongside the VB100 comparative review – watch out for a bumper edition!).

ANTI-BOTNET OPERATIONS

Operating in collaboration with a number of online organizations, the Dutch National Crime Squad's High Tech Crime Team seized 143 command and control servers of the Bredolab botnet last month.

Also involved in the takedown effort were a Dutch hosting provider, the Dutch Forensic Institute, security company *Fox IT* and GOVCERT.NL (the Dutch computer emergency response team). The investigation also led to the arrest by Armenian police of an individual suspected to have masterminded the botnet.

Despite the seizure of the command and control servers though, a couple of command nodes were found to still be active a few days later – leading to suspicion that a second group of bot herders have begun to issue new instructions to the botnet. The Dutch authorities have indicated that their investigation of the botnet is ongoing.

Meanwhile, in the UK a joint operation between the Metropolitan Police and Finnish authorities culminated in a Scottish man pleading guilty last month to 'causing unauthorized modification to the content of computers' as part of his involvement in the m00p hacking group. The group infected tens of thousands of machines worldwide by sending malware attached to spam messages. Thirty-three-year-old Matthew Anderson's role was in distributing millions of spam messages.

According to the Metropolitan Police, Anderson took control of the infected computers, on occasion activating their webcams to spy on their owners. During the investigation, screen grabs were found on Anderson's computers taken from webcams as well as copies of private documents including wills, medical reports, CVs, password lists and private photographs.

Anderson was arrested in 2006 and will be sentenced later this month.

DC Bob Burls of the Police Central e-Crime Unit, who was involved in the m00p investigation, will be detailing what it is that makes botnets the Internet weapon of choice at the VB Seminar on 25 November in London. See http://www.virusbtn.com/seminar/ for details.

Prevalence Table – September 2010[1]

Malware	Type	%
Autorun	Worm	11.65%
VB	Worm	9.31%
Conficker/Downadup	Worm	6.69%
Heuristic/generic	Misc	6.06%
FakeAlert/Renos	Rogue AV	5.35%
OnlineGames	Trojan	3.55%
Delf	Trojan	3.53%
Heuristic/generic	Virus/worm	3.48%
Sality	Virus	3.16%
Adware-misc	Adware	3.11%
Downloader-misc	Trojan	3.10%
Injector	Trojan	3.08%
Zbot	Trojan	2.61%
Crypt	Trojan	2.30%
Agent	Trojan	1.94%
Virut	Virus	1.92%
Iframe	Exploit	1.87%
StartPage	Trojan	1.82%
Small	Trojan	1.51%
Exploit-misc	Exploit	1.48%
Alureon	Trojan	1.47%
AutoIt	Trojan	1.39%
Peerfrag/Palevo/Rimecud	Worm	1.27%
PDF	Exploit	1.19%
Bancos	Trojan	1.14%
Tanatos	Worm	1.06%
Hupigon	Trojan	1.04%
Dropper-misc	Trojan	0.94%
Vobfus	Worm	0.91%
LNK	Exploit	0.68%
Bifrose/Pakes	Trojan	0.67%
Themida	Packer	0.62%
Others[2]		10.06%
Total		100.00%

[1]Figures compiled from desktop-level detections.

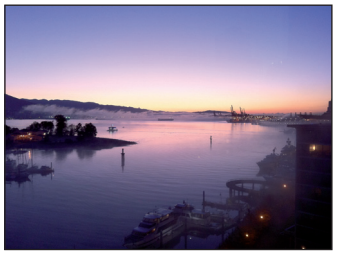
[2]Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

CONFERENCE REPORT

VANCOUVER EXPEDITION

Helen Martin

Last month saw the conclusion of VB2010 – marking the 20th anniversary of the Virus Bulletin conference – in Vancouver, Canada. It was *VB*'s second visit to Vancouver, the last sojourn having been for the ninth *VB* conference, in 1999. The vibrant, multicultural city of Vancouver, with its spectacular natural surroundings, frequently tops the charts as one of the best places to visit in North America, has been ranked among the top ten restaurant cities in the world, and has often been voted as one of the world's most liveable cities. With all that under its belt it's a wonder it took us 11 years to return!



The Westin Bayshore hotel provided the perfect setting for the anniversary celebrations, with stunning views across the bay to North Vancouver and along the seawall to Stanley Park. Even the weather gods smiled on the 20th birthday of the conference, generously giving us some beautiful Indian summer weather in late September (not that the conference team got to experience it, but the outside world looked nice as we peeked through the windows).

Whether it was the appeal of the beautiful city of Vancouver, the buzz surrounding Stuxnet, or simply the industry's recognition of the importance of getting together to share insight and knowledge, this year's conference exceeded all expectations in terms of delegate numbers, with a turnout of more than 360. In a period in which budgets are still tight as world economies begin the slow process of recovery we were thrilled to see such a large turnout – although the credit is surely due to the presenters and the papers on the schedule for creating such a draw.

THE OPENER

The conference kicked off on Wednesday morning with a keynote address by *Facebook* malware researcher Nick Biologorskiy. Nick revealed that the six-year-old social networking site – which has only been available to the public for the last three years – has over half a billion active users worldwide, and that 56% of these log into the site every day. With such a large user base and high profile, the site is a prime target for attackers and security is a high priority for the company. Nick detailed some of the many different types of attack targeting the site, and explained how the company's security team works to shut them down – revealing that the majority of attacks are unsuccessful thanks to behind-the-scenes security, and the ones the public sees are a very small percentage of what the security team deals with. He also described some specific attacks – disclosing that in 2009, the authors of the *Facebook*-targeting Koobface worm made around $1.8 million through their botnet and that, through their research, the *Facebook* security team has managed to uncover their identities (and pass that information on to the authorities).

Following the keynote address the conference programme split into its traditional two-stream format, with papers in both streams relating to bringing the perpetrators of cybercrime to justice. In the corporate stream Raymond Pompon – former undercover FBI agent – looked at some successes and failures in tracking down malware authors, highlighting some of the key problems and the methods that are used to investigate and prosecute them. Meanwhile, in the technical stream, *Panda Security*'s Pedro Bustamante spoke about the takedown of the Mariposa botnet and the arrest of its operators. *Panda Security* was part of the Mariposa Working Group, which was instrumental in the takedown of the botnet and the subsequent arrests. Pedro explained how the botnet was being operated, how its operators were so successful in turning it into one of the biggest botnets ever – at one point controlling close to 13 million computers and netting more than 20,000 euros per month – and how the investigation was carried out. However, he also highlighted the fact that insufficient cybercrime laws in the countries from which the botnet was operated may make it difficult to achieve successful prosecutions.

Kaspersky's Dmitry Bestuzhev posed the question 'How much do you cost?', referring to the black market price of digital data. Following a quick series of questions to the audience he calculated that, once infected, the details of

The VB conference wasn't the only one celebrating an anniversary – both Microsoft and G DATA also celebrated landmark birthdays.

the average *VB* conference attendee – email address, IM account, *Facebook* account, *PayPal* account, bank account and so on – would be worth $7,810.

Paul Baccas spoke about the heuristic detection of malicious PDFs, revealing that *SophosLabs* has seen an exponential rise in malicious use of PDFs, with nearly all drive-by web attacks containing a PDF component and a lot of infected PDFs also being emailed. Paul conducted a poll of the audience, asking whether PDF should be replaced with a safer format. An overwhelming majority of the audience agreed that it was time to retire the PDF. Paul also urged *Adobe* to remove JavaScript support from the format based on the results of his research.

Later in the technical stream, Donald DeBolt discussed the technical details behind black hat SEO attacks, explaining their logic flow, the abuse of *Google Trends* keywords, and identifying the technologies exploited. Dan Hubbard then drew the day to a close with a demonstration of how easy it is for criminals to contaminate real-time search results.

CHILD'S PLAY

The drinks reception at the end of the first day provided ample opportunity for delegates to relax and unwind – while several also took up the opportunity to regress into childhood. This year's drinks reception was dubbed the 'VB games night' as delegates were invited to roll back the years and rediscover their inner child with games ranging from the intellectual to the plain silly. Delegates were seen getting competitive over *Hungry Hippos*, *Operation*, *Buckaroo* and *Connect 4*, among others, while the intellectuals could be seen deep in concentration over a chess board or mastering *Othello*.

All the games used at the drinks reception were donated after the conference to a community project operating childcare programmes in the inner-city area of Vancouver.

Serious stuff – delegates show their competitive streak at the VB drinks reception and games night.

IN THE MIDDLE

On Thursday morning in the corporate stream, Gunter Ollmann discussed the limitations of current methods for measuring botnets and their associated malware components – highlighting the fact that botnet numbers are often overinflated.

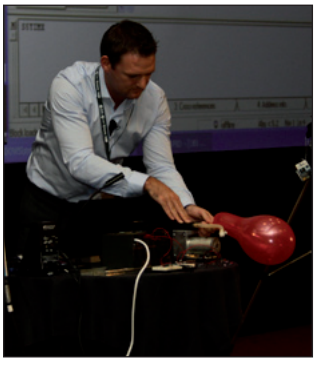
Next, Stefan Tanase investigated the role of social networks in automated targeted attacks, describing how criminals are using such attacks to get their foot in the door and go deep inside corporate networks by targeting a small number of specific employees using information gleaned from social networks.

After lunch, Carey Nachenberg and Vijay Seshadri analysed the real-world effectiveness of reputation-based security in detecting new malware. Having outlined the definition of reputation-based security and highlighted the differences between such an approach and cloud-based fingerprinting, they described *Symantec*'s reputation-based system and took delegates through an evaluation of the system, concluding that the reputation-based security approach provides a substantial independent layer of protection.

STUXNET, STUXNET, STUXNET

Most of the technical stream on Thursday was devoted to the last-minute papers – papers that had been submitted and selected just a couple of weeks before the start of the conference in order to allow up-to-the-minute content to be presented. This year's crop of last-minute papers were very strong, including presentations on smartphone dialers, ATM malware defences, exploit packs and the first 64-bit rootkit – but undeniably, the buzzword of the conference was Stuxnet.

Two last-minute presentations were devoted to the piece of malware everyone was talking about. First up, Liam O'Murchu disclosed some of the details he and the *Symantec* team have uncovered about the highly complex malcode that targets SCADA systems. To add drama to the proceedings he demonstrated, with the aid of an air pump and a confetti-filled balloon, how a Stuxnet-like piece of proof-of-concept code can override a programmable logic controller (PLC) to take control of a piece of



Liam O'Murchu demonstrates the world's most expensive balloon pump.

machinery. Liam hooked the PLC up to the air pump and programmed it to inflate the balloon for just five seconds. He then infected the PLC with his proof-of-concept code and set it to run again – but this time, rather than stopping after five seconds, the air pump continued inflating the balloon until eventually it burst, showering the front row of the audience with confetti and generating an excited round of applause. He suggested that, had the PLC been connected to, say, an oil pipeline, one could imagine how the results could have been significantly more destructive.

Next, a combined presentation from *Microsoft*'s Peter Ferrie and Holly Stewart and *Kaspersky Lab*'s Costin Raiu provided a discovery timeline for the malware – revealing that there is evidence that Stuxnet code dates back as far as January 2009 – as well as full details of the four zero-day vulnerabilities it uses.

In a break from tradition, a 50-minute panel-style question-and-answer session took place with the presenters of both talks after the two presentations. Even with the extension of the Q&A session, there wasn't enough time for all the questions the audience wanted to ask. The Stuxnet presenters also attracted much media attention, with interviews taking place in the press room almost every minute of the day, and film crews from both the CBC and the BBC attending to get the latest updates on the subject.

SONG, DANCE AND BIRTHDAY AWARDS

No *VB* conference would be complete without the traditional gala dinner evening and, in celebration of the 20th anniversary of the conference, this year's gala evening saw a special addition in the form of the VB2010 awards ceremony.

Chor Leoni and the Lorita Leung dancers add a little Canadian culture and colour to the evening.

And the winners are... (clockwise from top left): Peter Ferrie, the Spamhaus team, Andrew Lee presenting the best educator award to Mikko Hyppönen, and Righard Zwienenberg.

Diners entered the ballroom to the melodic tones of Chor Leoni men's choir who performed a selection of traditional Canadian folk songs dressed in their all-Canadian hockey shirts. Later we were treated to a visual feast in the form of a performance by the Lorita Leung dance company – North America's leading Chinese dance troupe.

After the singing, dancing and dining it was on to the awards ceremony. Ten years ago, at VB2000 in Orlando, an award was given to the individual considered to have contributed the most to the AV industry in the first ten years of the *VB* conference. To celebrate the 20th anniversary of the conference, the organizers decided to revive that award, along with five new awards that recognize the tremendous work and achievements of individuals in the industry.

The nominations were all made by visitors to the *VB* website and conference delegates voted during the first two days of the conference to decide the winners in each category. The winners were as follows:

Greatest contribution to the anti-malware industry in the last ten years – *Peter Ferrie*

Greatest contribution to the anti-spam industry in the last ten years – *The Spamhaus Project*

Best newcomer to the anti-malware industry in the last ten years – *Pierre-Marc Bureau*

Best educator in the anti-malware industry – *Mikko Hyppönen*

Most innovative idea in the anti-malware/spam arena in the last ten years – *Righard Zwienenberg*

Lifetime achievement award for services to the anti-malware industry – *Eugene Kaspersky*

Without exception the winners were very popular choices with the audience. Although there wasn't enough time for speeches from all the award winners, Righard Zwienenberg (winner of the award for most innovative idea, for *Norman*'s *Sandbox*) made a special request to say a few words in acknowledgement of Kurt Natvig's enormous contribution to the development of the *Sandbox*. Our best wishes go to Kurt, who wasn't able to attend the conference.

My thanks go to Eddy Willems, Paul Baccas, Randy Abrams, Andrew Lee, Richard Ford and Paul Ducklin for their help in introducing and presenting the awards. Each of them did such a professional job that you'd think they were regulars at red carpet events.

THE END IN SIGHT

Terry Zink kicked off the final morning of the conference in the corporate stream with a look at the psychology of spamming, looking at the role our emotions play when evaluating the content of a spam message and how this works to the advantage of the spammer. David Koconis of *ICSA Labs* then presented an overview of the certification body's anti-spam test methodology, highlighting some of the differences between its methodology and that of other anti-spam tests.

The technical stream saw Thomas Dullien challenging conventional wisdom on byte signatures – arguing that byte signatures are not inherently bad – and Catalin Cosoi discussing the benefits and the downsides of scanning URLs in the cloud.

Later in the afternoon, Pierre-Marc Bureau and Joan Calvet described a Canadian government-funded collaboration between academic researchers and *ESET* to conduct a large-scale malware experiment. The researchers used a computer cluster to boot virtual machines, infect them with malware and let them connect together as a botnet, which they were then able to study and experiment with.

To round off proceedings on Friday afternoon a panel of experts – Lysa Myers, Andrew Lee, Catalin Cosoi, David Perry and Nick Bilogorskiy – were led by Mikko Hyppönen in a discussion of social networking and computer security.

Setting the scene for the discussion, Mikko set up a *Twitter* wall using the hashtag #vb2010 so that, as the discussion progressed, tweeted comments and questions from the audience (both present and remote) would appear live on screen. (Thankfully this was a few days before Mikko was banned from *Twitter*!) Several serious topics were addressed, including the issue of compromised and fake accounts – with perfect illustration of how easy it is to impersonate others online provided by onscreen tweets from 'VesselinBontche' and a Barack Obama impersonator. While the first tweet from the Vesselin impersonator – 'firstly, social networks are for idiots' – had the audience wondering if we had been joined remotely by the great man himself, subsequent tweets had a distinctly more fishy ring to them. At the end of the session Mikko concluded that 1. you can't trust anything on the Internet, and 2. social networks can be awfully distracting – he also warned that since social networks seem to be very much here to stay, the security industry must be prepared for more attacks.

At the cutting edge: the VB2010 speakers.

THANK YOUS

There is never enough space in these reports to mention more than a small selection of the speakers and presentations at the conference, and I would like to extend my warmest thanks to all of the VB2010 speakers for their contributions, as well as to the sponsors of the event: *Avast Software*, *ESET*, *K7 Computing*, *CA*, *Kingsoft*, *Microsoft*, *MX Tools*, *ArcaBit*, *OPSWAT*, *Sunbelt Software*, *TrustPort*, *Beijing Rising* and *AVIEN*.

Next year the conference lands in sunny Spain with the event taking place 5–7 October 2011 at the Hesperia Tower hotel in Barcelona. I very much look forward to welcoming you all there.

Photographs courtesy of: Andreas Marx, Eddy Willems, Pavel Baudis, Tiffini Schwarzkopf and Tjark Auerbach. More photographs can be viewed at http://www.virusbtn.com/conference/vb2010/photos/ and slides from the presentations are available at http://www.virusbtn.com/conference/vb2010/slides/.

MALWARE ANALYSIS

DEELAED LEARNING

Peter Ferrie
Microsoft, USA

Not long ago, a new virus writer appeared, using the name 'hh86'. Rumour had it that hh86 was female – a rarity in the virus-writing world. There was a flurry of activity from hh86 over a period of about three months, producing a handful of viruses using new techniques, and then... she was gone without a word. The model virus writer perhaps.

At first glance, I thought that her first virus (Deelae.A) was simply a copy of a virus created by the virus writer roy g biv. A slightly closer look revealed some novel size optimizations (as well as some opportunities that were missed, and some 'optimizations' that are the same size but slower to execute) as well as some differences in style. It's clear that hh86 was 'inspired' by roy g biv's work. In Hollywood, they'd call that 'reimagining'.

TECHNOLOGY ANTHOLOGY

Let's start with the things that are the same. The most obvious is the construction of a Structured Exception Handler (SEH) which is used as a single point of exit when an error occurs. Even the exception condition is the same – an interrupt 3 instruction. One annoying technique the pieces of malware have in common is the loading of the stack with as many parameters as possible, prior to calling APIs in sequence (see *VB*, October 2004, p.4, for an illustrated example).

Another technique they share is the method used to retrieve a DLLBase value from the PEB_LDR_DATA structure in the Process Environment Block. This technique was actually first published by the virus writer Ratter, and he seems to be the more likely source here. The reason for this speculation is that roy g biv has used the technique only once – it wasn't in a virus at all (it was in the BASLR tool), and it loaded a different DLL from the one that hh86 is trying to load. That difference introduces a problem for hh86.

The PEB_LDR_DATA structure contains three structures ('InLoadOrderModuleList', 'InMemoryOrderModuleList', and 'InInitializationOrderModuleList', but *Microsoft* documentation lists the first structure as 'reserved', names the second one, and implies that the third does not exist). Any of the three structures can be parsed to find the list of DLLs that are loaded. The difference is in the order of the pointers that are referenced by the structures, and thus the amount of code used to complete the retrieval.

All three virus writers used the InInitializationOrderModuleList structure (although roy g biv called it the 'InLoadOrderModuleList'). This results in the smallest code, but it does not work on *Windows 7* when it is used to retrieve the DLLBase of kernel32.dll. roy g biv's BASLR tool loaded ntdll.dll, which does work on *Windows 7*, but hh86 (and Ratter) loaded kernel32.dll. Thus, Deelae.A and Deelae.B do not work on *Windows 7*. This might be considered the first bug, and hh86 seems to have thought so too, since it was fixed in Deelae.C.

IMPORT-ANT DETAILS

The code used to resolve the imports is also similar, up to a point. Deelae uses the CRC32 method, to avoid the need to store the strings. However, the CRCs are not sorted according to the alphabetical order of the strings they represent, so multiple passes over the export table are required to resolve the imports. The code that resets the index contains one potential problem. The code that checks for the end of the list contains several potential problems.

We'll start with the problems in the list termination code. The first problem is that only one byte is checked (this is especially curious, given that four bytes were allocated in Deelae.A and Deelae.B). This prevents the use of any function whose CRC32 value would have the checked value in the lowest byte.

The second problem is that the checked value is not constant in Deelae.A and Deelae.B, thanks to the way in which it is constructed. Instead, it depends on the previous CRC32 value. If the top byte of the previous CRC32 value is less than 0x80, then the checked value will be zero. Otherwise, the checked value will be 0xff. As a result, the list cannot have the form 0x00-7f x1 x2 x3 0x00 y1 y2 y3, or 0x80-ff x1 x2 x3 0xff y1 y2 y3, because in each case, the list will appear to terminate too soon.

The third problem is that if the terminating value is 0xff, it leads to a real bug in Deelae.A and Deelae.B. A few instructions later, some space is allocated on the stack using a modification to the register that held the checked value. If the terminating value is 0xff, then the stack pointer is moved in the wrong direction, destroying the SEH registration, and leading quickly to a crash.

This third problem was fixed in Deelae.C in two ways. The first was by changing the comparison register to one that always holds a zero – but the byte-check remains, and therefore so does the first problem. This is despite the fact that the comparison register is entirely zero, and thus all

four bytes could have been checked. However, in Deelae.C, the CRC table was changed to end with a single byte, and this change is also present in Deelae.D and Deelae.E. The second way was to allocate the stack space using an instruction specifically designed for the purpose, so no arithmetic overflow can occur.

The problem with the index reset code is that the index might be set to 0xffffffff instead of zero, due to the second problem above. This index is used to look inside the Name Pointer table. In the (obviously impossible) case that the Name Pointer table had a Relative Virtual Address (RVA) of zero (that is, pointing to the 'MZ' part of the file header), the attempt to retrieve the first name RVA would access memory outside of the image, and cause an exception. However, the exception would be intercepted and the virus code would simply exit without issue. There is also the remote possibility that a wanted string appears immediately before the import table, and that would lead to the wrong function pointer being retrieved. That could result in unexpected behaviour, but a crash seems likely. The exception should be intercepted here too, but depending on what is called, a real crash might occur. The file to infect might also be corrupted.

DIFFERENCE OF OPINION

One of the major ways in which hh86's viruses differ from those of roy g biv is in the file handling. roy g biv's viruses would not infect files that are protected by the System File Checker (this feature was added in Deelae.F), they would remove the read-only attribute if required, and they would not infect files that were for a different CPU, or were DLLs if the virus did not support them.

In contrast, hh86's viruses do not care about the read-only attribute, they do not check the CPU, and they will infect DLLs (in the unlikely event that they are misnamed). They do, however, clear the 'NX' bit if it is set. This allows the viruses to execute on systems with Data Execution Prevention enabled, even if the section is not marked as 'executable'. roy g biv's viruses, on the other hand, set the 'executable' bit in the section header. Both techniques achieve the same goal. roy g biv's viruses search recursively through all directories, and infect files regardless of their filename; hh86's viruses search only within the current directory, and only for '*.exe'. hh86's viruses are also aware of SafeSEH (see below), which did not exist when the majority of roy g biv's viruses were created, but even his most recent viruses do not support it. Interestingly, neither Deelae.A nor Deelae.B is aware of the NO_SEH bit (see below), but Deelae.C, Deelae.D and Deelae.E are.

Now it's time to describe the specifics of each of hh86's viruses.

CERTIFICATE HOLDER

Deelae.A checks if the Certificate Table data directory indicates the presence of a digital certificate of at least 4KB in size after the end of the last section. If one is found, the virus zeroes the Certificate Table data directory (because it will be overwritten) and the Load Config Table to disable SafeSEH. Disabling SafeSEH allows the virus to raise exceptions without causing the application to be terminated by the operating system *if the NO_SEH bit is also clear*. The reason for this is because the NO_SEH bit states that no code within the image uses SEH, and thus no code is allowed to call SEH within the image. This bit was introduced in *Windows XP* SP2, but was not documented until recently.

The virus increases the virtual and physical sizes of the last section, and the SizeOfImage value, by 4KB. It copies itself over the certificate data, and then saves and alters the host entrypoint to point to the start of the certificate data. Finally, the virus copies itself over the certificate data and then forces an exception to occur, thus ending the infection process for that file. In the past, few files carried digital certificates, which would have limited the number of available candidates for this virus. However, an increasing number of files are now released with digital certificates, so overwriting the certificate has become a viable technique to avoid an increase in file size.

EXPECT DEELAES

Deelae.B is identical to Deelae.A with the exception of the entrypoint hook. This time, the virus is interested in the Delay Import Descriptor table.

The Delay Import Descriptor table is used, as the name implies, to delay the importing of DLLs until they are needed. That can improve the start-up time for some applications, and also reduce the memory requirements. If a particular code-path is the only one that requires a certain DLL, and if that code-path is not taken, then the DLL will not be loaded unnecessarily. The table has been documented for a long time, but incorrectly, despite several revisions to the documentation. Specifically, the 'Attributes' field is documented as 'Must be zero', and 'As yet, no attribute flags are defined. The linker sets this field to zero in the image.' However, this is not true. The linker sets this field to 1, indicating that the table is valid. If bit zero is not set in the table, then an exception is raised by the application.

The virus retrieves the RVA of the Delay Import Descriptor, and checks that it points within the first section. The virus then retrieves the RVA of the Delay Import Address Table and checks that it points within the second section. If both checks pass, then the virus hooks the first address in the Delay Import Address Table to point to the start of the certificate data.

This entrypoint-obfuscating technique is new. roy g biv showed that the Bound Import Table can be hooked, though he went to extremes to hide the code (see *VB*, December 2006, p4). hh86 showed that the Delay Import Table can be hooked, and in multiple ways (see below), but she did not attempt to hide it (however, the popular tool known as the *Interactive Disassembler* does it for her – it hides the entrypoint in an attempt to avoid bad links in corrupted files).

FURTHER DEELAES

Deelae.C and .D are based on Deelae.B, but with some minor changes, and one major change. One minor change relates to the file mapping. Both Deelae.A and Deelae.B overwrote the certificate data, and thus mapped the file according to its original size. Deelae.C and Deelae.D increase the file size by 4,213 bytes, even before checking if the file is a candidate for infection. Of course, if an error occurs, then the viruses restore the file to its original size.

The major change is that the viruses will not infect files that have appended data. This is the infection marker – a technique that roy g biv has used for most of his viruses.

The viruses increase the virtual and physical sizes of the last section, and the SizeOfImage value, by 4KB, and then append themselves to the image (perhaps hh86 feels that there are not enough candidates for infection). The file size is increased permanently at this point by 4KB+1, which introduces a potential problem later. The viruses retrieve the RVA of the Delay Import Descriptor and check that it points within the first section. The viruses retrieve the RVA of the Unload Delay Import Table and check that it also points within the first section. If both checks pass, then the viruses hook the first address in the Unload Delay Import Table to point to the virus body. The problem here is that if either pointer is not within the expected range, then the file remains marked as 'infected', and contains the virus body, but there is no pointer to the code. This will be quite a common occurrence, since the Unload Delay Import Table is not used particularly often, and so the pointer will be null.

If the file is infected successfully, then the viruses attempt to zero only the Load Config Table (the Certificate Table data directory is no longer zeroed). However, there is a bug in Deelae.C which is that the destination of the write is a critical field within the section table instead of inside the data directories. As a result, files infected successfully by Deelae.C will often be corrupted. This bug was fixed in Deelae.D.

YET MORE DEELAES

Deelae.E is based on Deelae.D, but with two minor changes. One change is the field that is used for hooking. Instead of the Unload Delay Import Table, the virus uses the Bound Delay Import Table if the Time Stamp field is non-zero. The virus hooks the first address in the Bound Delay Import Table to point to the virus body.

The other change is the introduction of a bug. Previously, the host entrypoint was stored as an RVA, and adjusted dynamically. This allowed the virus to work with files that had Address Space Layout Randomization (ASLR) enabled. However, since the addresses in the Bound Delay Import Table are Virtual Addresses (VAs), hh86 seems to have assumed that it is acceptable to replace a VA for an external file with a VA for the infected image. This does not work if ASLR is enabled, unless a corresponding relocation item is included, because the host image can move, even if the referenced DLL does not. Note that roy g biv's Bounds viruses have the same problem.

MAXIMUM DEELAES

Deelae.F is based on Deelae.E, but with the combination of hooking methods from Deelae.E, Deelae.C/.D, and Deelae.B, in that order. That is, if the Time Stamp field is non-zero, then the virus uses the Bound Delay Import Table. If the Time Stamp field is zero, and if the Unload Delay Import Table is non-zero, then the virus uses that table. Otherwise, the virus uses the Delay Import Address Table. As before, the first address in the selected table is hooked, and the original address is stored in the virus body. A jump instruction is modified in the virus body to convert the address to a VA if that is required.

Deelae.F also makes use of the FPU to copy some pointers. This is intended to reduce the code size, but the implementation is flawed, and so the resulting code is larger than if the FPU had not been used at all.

CONCLUSION

Five new viruses and four kinds of entrypoint in three months. hh86 was the 'Energizer bunny' of the virus-writing community. It's a good thing that her batteries ran out.

TUTORIAL

EXPLOIT IDENTIFICATION

Mark Davis, USA

My previous articles (see *VB*, April 2010, p.21, May 2010, p.17 and August 2010, p.8) have introduced exploit frameworks like Fragus, Tornado, and many others, and described how to analyse them using LAMP/WAMP servers. This article walks through a Tornado kit, start to finish, showing the process required to identify exploits in the kit. Principles from this example are applicable to the research of all such exploit frameworks. It begins with script or netflow analysis, decoding, more analysis, and continued correlation and testing, until reasonable confidence for exploit identification is acquired.

APPROACHING THE KIT

The first step is awareness of the kit. An analyst may perform multiple queries and coordinate in both public and private arenas to get an idea of what is already known about a kit. This can greatly expedite research angles and context for a researcher when analysing a kit. For example, an analyst may get an idea of how long a kit has been in the wild, the exploit vectors expected and/or deep kit analysis performed by others, and more.

For this demonstration a copy of the Tornado exploit kit was captured in the wild. A few directories exist along with a few files at the root level of the kit:

- Data/
- Exploits/
- Include/
- Stats/
- .htaccess
- Count.php
- Dump.sql
- Getexe.exe

Familiarity with the context of exploit kits (see previous articles) helps the analyst to assume the following about each element of the kit:

- **Data/** Contains possible log files for the kit itself, stolen data, or support media for the kit.
- **Exploits/** Probably contains exploits, but if this is a demo version, only a few common vectors will be present (demos usually exclude the important exploits).
- **Include/** Contains elements required for the kit set-up, normally including *MySQL* configuration, *GeoIP*, crypting, and similar configuration files.
- **Stats/** Contains statistics related to the kit, used to display in the kit (e.g. number of infections per country).
- **.htaccess** This is probably an *Apache* distributed configuration file used to control access to the kit when on a web server.
- **Count.php** This is probably a PHP file that is used to track something.
- **Dump.sql** This is probably a sample *SQL* database file used in a demonstration of the kit or possibly containing full abuse data.
- **Getexe.exe** This is probably the payload for the kit and what will be seen in URLs when exploitation is successful.

To identify the exploits the analyst immediately navigates to the exploits directory and finds files named ‘x1.php’ through ‘x16b.php’. This is a sequential naming convention that suggests that exploits are carefully managed by a unique number and/or letter variant. An analyst that is paying attention to this pattern will realize that online abuse data may point to other exploits, like 17.php or others not found in the demonstration kit. If this is the case, the analyst can work with the demonstration exploits and then correlate abuse data to suspected vectors of exploitation for 17.php and above to obtain a very solid concept of what the kit is confirmed to exploit and its likely exploits. This also gives the analyst the ability to configure behavioural environments to perform live tests against new Tornado exploit kits to confirm suspected exploit vectors for 17.php+ and higher.

Inspection of the content of the exploit files should first take place inside *Notepad* ++ or other similar safe viewing utility. All files contain the same ‘Zend’ header data and obfuscated content as shown in the snippet below from x1.php:

```
Zend
2006022801 2 0 3 1477 3349 xùŸ2Wmo£FŽTU:9R‰Â‰ò¡Á²—îæ¥
>Nâõúár9Ån¯jÓÖw¬sq£|éÏí¯è,`»é%ªjÛvž™ }
```

At this point the analyst realizes that deobfuscation of the data is required before analysis, but may not understand the ‘Zend’ header. It is clear that all files are ‘Zended’, so a *Google* query may help to clarify this. *Google* queries such as ‘zend header’, ‘zend php files’ or ‘zend obfuscation’ may reveal content to help the analyst understand what he is dealing with and how to deobfuscate it. In this example, the analyst probably finds zend.com rather rapidly and learns of a commercial solution for working with PHP management and code. Next, a more descriptive *Google* query like ‘zend php file obfuscation’ is appropriate, leading to pages that discuss obfuscated PHP code and how to decode such files. Within a few minutes the analyst is able to understand the origin of Zend header files and that there are a variety of tools that can be used to deobfuscate such ‘zended’ scripts.

Several utilities exist online to de-zend scripts, such as http://old.boem.me/dezend/. However, analysts should never blindly trust any such utility, and should only use them inside a safe lab or virtualized environment rather than on a production machine. Some tools require terminal line interaction while others are GUIs, but eventually a tool can be found that successfully decodes the obfuscated PHP files. In this case, de-zending tools and success may vary based on the version of PHP being worked with, such as PHP4 or PHP5. Trial and error may be required to eventually find a successful vector for deobfuscating the code.

Now, a copy of the files exists on the analyst's machine, de-zended and in the clear. x1.php now has introductory content as shown in the snippet below:

```
<?php
/*********************/
/*           */
/* Dezend for PHP5 */
/*      NWS     */
/*   Nulled.WS  */
/*           */
/*********************/

if ( defined( "GRANTED" ) )
{
  exit( );
}
echo "var exeurl=url+'1';\nfunction CreateO(o,n)
```

The first part of this script contains a header injected by the de-zending tool. The important part is the 'if' statement and below, which clearly shows hostile JavaScript. At this point the analyst may quickly scan the document for important clues such as CLSID values, eval statements, strings that may be unique to an exploit, or strings used by the actor that may reveal the identity of the exploit. When performing this kind of visual review of a script, analysts should use *Notepad* ++ or a programming package so that line numbers and colour-coding of the elements can be viewed. This greatly aids in reviewing data when compared to *Notepad* viewing. An example of this is shown in Figure 1.

In reviewing x1.php de-zended scripts, we can see that multiple strings exist in the document, providing clues to possible exploit functionality:

- ADODB.Stream
- BD96C556-65A3-11D0-983A-00C04FC29E36
- BD96C556-65A3-11D0-983A-00C04FC29E30
- AB9BCEDD-EC7E-47E1-9322-D4A210617116
- 0006F033-0000-0000-C000-000000000046
- 0006F03A-0000-0000-C000-000000000046

Figure 1: Colour coding in Notepad ++ helps the analyst.

- 6e32070a-766d-4ee6-879c-dc1fa91d2fc3
- 6414512B-B978-451D-A0D8-FCFDF33E833C
- 7F5B7F63-F06F-4331-8A26-339E03C0AE3D
- 06723E09-F4C2-43c8-8358-09FCD1DB0766
- 639F725F-1B2D-4831-A9FD-874847682010
- BA018599-1DB3-44f9-83B4-461454C84BF8
- D0C07D56-7C69-43F1-B4A0-25F5A11FAB19
- E8CCCDDF-CA28-496b-B050-6C07C962476B
- BD96C556-65A3-11D0-983A-00C04FC29E36

An Internet search for possible exploits and/or exploit examples related to the above strings and CLSID values can now be undertaken by the analyst. Unique to this first example is the large number of CLSID values and the string 'ADODB.Stream', which is not common among exploit files (most contain just one to three such strings). By combining terms and looking for exploits, the analyst can run the following query on *Google*: 'adodb.stream BD96C556-65A3-11D0-983A-00C04FC29E36 exploit'. The first result from this query refers to an MDAC MS06-014 exploit:

```
Internet Explorer (MDAC) Remote Code Execution
Exploit (MS06-014 ...
DataSpace', '{BD96C556-65A3-11D0-983A-
00C04FC29E36}'], .... var s = CreateO(a, "WScript.
Shell"); var o = CreateO(a, "ADODB.Stream"); var e =
s. ...
securityreason.com/exploitalert/975 -
```

Browsing the first page of search results reveals lots of information about the MDAC vulnerability, articles on attacks in the wild using the MDAC vulnerability, several behavioural analysis and anti-virus reports related to the

vector and strings queried, and exploit files used by bad actors to exploit the MDAC vulnerability. If an analyst is not familiar with this exploit, each of these leads can be followed up until reasonable certainty is obtained as to the identity of the exploit. This often involves a few security reports followed up by a Milw0rm or Metasploit script analysis to accurately identify the structure and context of exploits compared to the file under analysis.

Conclusive identification of an exploit can only take place with the following actions taken after initial research is performed:

- An exact copy of a known identified exploit online matches that of the exploit being analysed.
- A minor copy of an exploit is identified, with no major changes in core functionality of the exploit vector.
- Carefully controlled behavioural analysis of a specific exploit vector is employed against the suspected vector inside a LAMP/WAMP server or against a remote live server. This may involve fully patching a system and then removing the patch suspected to be the fix for the vector being targeted by the exploit file.
- An expert in exploitation analysis qualifies the initial findings.

Another item that analysts should look for when performing kit analysis is the bad actor's comments and marketing media. The authors of exploit kits often use slang when referring to specific common vectors of attack, such as 'MDAC, Snapshot, qt' and so on. Learning the common slang terms used can serve as a pointer to an analyst investigating an exploit script within a kit.

In reviewing the de-zended Tornado scripts, many hours may pass before key elements of each script are identified, researched, correlated, and/or confirmed. When done with such research it is common to have some vectors of exploitation that have been identified conclusively, while others are found to be highly likely, and others still may be unconfirmed but highly likely based upon both local lab tests and correlation to patterns and remote data that suggest full functionality of a kit. In the case of Tornado, the following exploit vectors can be identified in the aforementioned PHP files:

x1: CVE-2006-0003. Microsoft Windows Server 2003 Service Pack 1 RDS.Dataspace ActiveX Control Access Control Vulnerability (Microsoft Data Access Components – MDAC)

x2: CVE-2006-3730. WebViewFolderIcon (WVF)

x3: CVE-2007-0024. Vulnerability in Vector Markup Language Could Allow Remote Code Execution (929969) (VML)

x4: CVE-2007-0015. Buffer overflow in Apple QuickTime 7.1.3

x5, x6: CVE-2006-0005. Microsoft Windows Media Player Plugin Buffer Overflow Vulnerability (WMP Plugin for Opera/FireFox Embed).

x7, x7b: CVE-2007-6166. QuickTime RTSP Response vulnerability

x8: CVE-2006-6884. WinZip FileView ActiveX controls CreateNewFolderFromName() Method Buffer Overflow

x9: CVE-2007-2987. Zenturi ProgramChecker ActiveX (sasatl.dll) Remote Buffer Overflow

x10: CVE-2007-3147, CVE-2007-3148. Yahoo! Webcam view Utilities ActiveX Control Vulnerable to Arbitrary Code Execution

x11: CVE-2009-1930. Microsoft Windows Server 2008 Service Pack 2 Telnet Server Unspecified Vulnerability (Opera 9.25 and earlier; TN3270)

x12: CVE-2006-5745. Vulnerability in Microsoft XML Core Services Could Allow Remote Code Execution (928088)

x15, x15b: CVE-2003-0111. Java ByteCode Verifier / Flaw in Microsoft VM

x16, x16b: CVE-2007-0038. Microsoft Windows Animated Cursor Remote Code Execution Vulnerability (925902) (ANI) Vulnerability in Microsoft Management Console Could Allow Remote Code Execution (917008; MS06-044). Publicly reported but not confirmed in lab samples: CVE-2006-3643.

Once research has been completed the analyst can perform follow-up kit analysis by tracking common strings, CLSID values and other components that led to a successful identification of an exploit vector. This greatly expedites future kit analysis since many of the vectors used in a kit are widely used by many kits. As such, once the steep learning curve of kit analysis has been completed the analyst will be able to identify new kits easily and rapidly, and more importantly, identify new exploit vectors used by a kit in the wild. As an example, common slang terms like 'TN3270' or 'TN 3270' are commonly used to refer to a *Telnet* server vulnerability 'Microsoft Windows Server 2008 Service Pack 2 Telnet Server Unspecified Vulnerability (Opera 9.25 and earlier; TN3270)', CVE-2009-1930, MS09-042.

To apply what you have learned in this article try to identify the exploit using this CLSID: 10072CEC-8CC1-11D1-986E-00A0C955B42E. You should be able to get an idea of what the exploit vector is within 15 seconds or less, tied to an exploitation that first began in 2006 and 2007 in the wild.

TECHNICAL FEATURE

ANTI-UNPACKER TRICKS – PART FOURTEEN

Peter Ferrie
Microsoft, USA

New anti-unpacking tricks continue to be developed as older ones are constantly being defeated. This series of articles has described some tricks that might become common in the future, along with some countermeasures [1–14].

In this final article of the series we look at anti-unpacking by anti-emulating.

Unless stated otherwise, all of the techniques described here were discovered and developed by the author.

1. SOFTWARE INTERRUPTS

1.1 Interrupt 0x2E

On *Windows XP* and later versions (but only on 32-bit platforms), if the CPU supports the SYSEXIT instruction, *Windows* will return in the EDX register the address of the next instruction to execute.

Example code looks like this:

```
    ;any value will work
    ;but requires user32.dll loaded
    or   eax, -1
    int 2eh
l1: cmp edx, offset l1
    jne being_debugged
```

The reason for this is obscure. This disassembly shows more:

```
    test d [esp+4], 1 ;ring check
l1: jne  l2 ;taken if ring 3
    ...
l2: iretd ;return to caller
l3: test b [esp+9], 1 ;check T flag
    jne  l2 ;use iret if set
    pop  edx ;edx=eip
    add  esp, 4 ;discard cs
    and  b [esp+1], -3 ;clear I flag
    popfd ;load flags
    pop  ecx ;discard error code
    sti
    sysexit ;fast return to caller
```

In this disassembly, there is no reference to either l1 or l3. However, what cannot be seen here is that code exists elsewhere in the kernel, which checks for CPU support for the SYSEXIT instruction. If such support exists, then the kernel adjusts the value at l1+1 such that the branch reaches l3 instead of l2.

When active, the only way to reach l2 is if the T flag is set. In all other cases, the faster SYSEXIT instruction is used instead of the IRETD instruction. As a side effect of that change, the EDX value always contains the EIP value on return.

Interestingly, *Windows 2000* contains similar code, as we can see in this disassembly:

```
    ;check for SYSEXIT support
    ;internal flag, not CPUID value
    test d [xxxxxxxx], 1000
    je   l1 ;taken if not supported
    test d [esp+4], 1 ;ring check
    je   l1 ;taken if ring 0
    ...
    pop  edx ;edx=eip
    add  esp, 8 ;discard cs, eflags
    pop  ecx ;discard error code
    sti
    sysexit ;fast return to caller
l1: iretd ;return to caller
```

Here, a variable is checked instead of using an altered branch. It has poorer performance, but it avoids the in-memory patch. However, the code that queries the CPU capabilities does not contain any code to enable this feature. As a result, the SYSEXIT path is never reached.

There is an additional unexpected behaviour in the 32-bit version of *Windows Vista* and later *Windows* versions. If the value in the EAX register exceeds the size of the standard service table, then *Windows* will call through the ntdll KiUserCallbackDispatcher() function, which in turn calls through the PEB->KernelCallbackTable table. The index that is used depends on the *Windows* version. For *Windows Vista*, the index is currently 0x4c, and for *Windows 7*, the index is currently 0x4a. These values could change in the future, but it is trivial to fill a table that can support any value. This technique could be used to redirect execution in an obfuscated manner for those platforms.

Example code looks like this:

```
    call GetVersion
    cmp  al, 5
    jb   l1 ;not Vista+
    push offset l2
    call GetModuleHandleA
    push offset l3
    push eax
    call GetProcAddress
    xchg ecx, eax
    jecxz l1 ;not supported
    push eax
    push esp
```

```
    push -1 ;GetCurrentProcess()
    call ecx
    pop  ecx
    loop l1 ;taken if not WOW64
    mov  eax, fs:[ecx+30h]
    mov  d [eax+2ch], offset l4
    int  2eh
    jmp  being_debugged
l1: ...
l2: db   "kernel32", 0
l3: db   "IsWow64Process", 0
l4: dd   4ah dup (0)
    dd   offset l1 ;Windows 7
    dd   0
    dd   offset l1 ;Vista
```

2. OPERAND-SIZE OVERRIDE

The operand-size override (0x66) can be used on instructions that transfer control. The result is that the EIP register is truncated to a 16-bit value. Execution resumes (if possible) from the resulting address.

Example code looks like this:

```
    xor  ebx, ebx
    push 40h
    mov  eax, esp
    push 3000h
    push esp
    push ebx
    push eax
    push -1 ;GetCurrentProcess()
    call NtAllocateVirtualMemory
    xchg ecx,eax
    db   66h
    jecxz l1
l1: ...
```

In this example, execution continues at the address (l1&0xffff). This technique works with all types of branch – the 7x form and the 0f xx form.

Example code looks like this:

```
    xor  ebx, ebx
    push 40h
    mov  eax, esp
    push 3000h
    push esp
    push ebx
    push eax
    push -1 ;GetCurrentProcess()
    call NtAllocateVirtualMemory
    test eax, eax
    db   66h
    je   l1
l1: ...
```

In this example, execution continues at the address (l1&0xffff). This technique also works with relative calls and relative jumps.

Example code looks like this:

```
    xor  ebx, ebx
    push 40h
    mov  eax, esp
    push 3000h
    push esp
    push ebx
    push eax
    push -1 ;GetCurrentProcess()
    call NtAllocateVirtualMemory
    call l1 ;determine eip
l1: pop ax ;discard low 16 bits
    call small l2
l2: ...
l3: ...
```

As with the previous example, execution continues at the address (l1&0xffff). However, unlike the previous example, this one can return to l3, with a balanced stack, simply by executing a 32-bit RET instruction.

Note the explicit mention of a '32-bit RET instruction'. This is important because the technique also works with all types of return (near and far).

Example code looks like this:

```
    xor  ebx, ebx
    push 40h
    mov  eax, esp
    push 3000h
    push esp
    push ebx
    push eax
    push -1 ;GetCurrentProcess()
    call NtAllocateVirtualMemory
    push small offset l1
    db   66h
    ret
l1: ...
```

As in the last example, execution continues at the address (l1&0xffff). Finally, this technique also works with the IRET instruction.

Example code looks like this:

```
    xor  ebx, ebx
    push 40h
    mov  eax, esp
    push 3000h
    push esp
    push ebx
    push eax
    push -1 ;GetCurrentProcess()
```

```
    call NtAllocateVirtualMemory
    pushfw
    push small cs
    push small offset l1
    iretw
l1: ...
```

As with the previous example, execution continues at the address (l1&0xffff).

Since this is a most uncommon use of the operand-size override, it is possible that some emulators will not support it.

3. MULTI-TASKING

The CPU supports the running of multiple tasks. Each of those tasks has access to various resources such as the CPU registers and the FPU. However, when a task switch occurs, the CPU saves only the CPU registers and none of the FPU state. Instead, the CPU sets the 'TS' (Task Switched) bit in a control register, which signifies that a task switch has occurred. Whenever the CPU encounters an FPU, MMX, or SSE instruction (with a few exceptions), it checks the state of that bit. If the bit is set, then the CPU checks the state of the 'MP' (Monitor Processor) bit. This bit is under software control. If it is also set, then the CPU raises an 'NM' (Non-Maskable) exception that refers to the co-processor. The software-based task manager intercepts that exception and saves the state of the FPU, MMX and SSE environment prior to clearing the TS bit to avoid a redundant save. The reason the MP bit exists is to avoid the relatively large overhead of saving the FPU state in the event that it is entirely unnecessary because a task did not use the FPU at all. There is also the possibility that several related tasks might share the FPU. In such a case, the task manager can also clear the MP bit to avoid a redundant save.

The task-switching behaviour can be exploited as an anti-emulation trick. Specifically, a process can execute an FPU instruction, thus causing the NM exception to be raised and the FPU state to be saved. The task manager will clear the TS bit in response to this event, and potentially clear the MP bit too. After some time passes and other tasks are executed, the task manager will set the MP bit again if it was cleared, and the processor will set the TS bit again. This cycle will continue until eventually the process resumes execution. At that time, the two bits should be set. A process can detect this cycle.

Example code looks like this:

```
    wait ;raise NM
l1: smsw ax
    and  al, 0ah
    cmp  al, 0ah
    je   l1 ;wait while TS and MP
l2: smsw ax
    test al, 2 ;wait for MP
    je   l2
    test al, 8 ;check for TS
    je   being_debugged
```

This technique is used by Waledac. However, it does not work on the 64-bit versions of *Windows*. Specifically, the loop at l2 never exits, because the MP bit is never set again for the process.

4. VirtualPC-SPECIFIC

There are some common methods in shellcode for finding the value of the EIP register using instructions that contain no bytes with a value of zero. One of those methods uses an FPU instruction.

Example code looks like this:

```
l1: fldz
    fnstenv [esp-0c]
    pop eax
l2: ...
```

When l2 is reached, the value in the EAX register will be the address of l1. Thus, given the following code, it seems reasonable to assume that the branch at l3 will never be taken:

```
l1: fldz
    fnstenv [esp-0c]
    pop eax
l2: cmp eax, offset l1
l3: jne being_debugged
```

However, this is an invalid assumption. In *VirtualPC*, single-stepping over the fldz instruction results in a completely different value in the EAX register. The cause is unknown at the time of writing, but the value appears to be a constant (0x74b036). This means that the code could be altered in a very subtle way.

Example code looks like this:

```
org 74b035h
l1: fldz
    fnstenv b [esp-0ch]
    pop eax
    dec b [eax+(offs l2-offs l1)-1]
    mov eax, offset l3+01000000h
l2: mov ecx, offset being_debugged
    jmp eax
l3: ;...
```

If the code executes freely, then execution continues from l3. However, single-stepping over the fldz instruction causes the 'mov ecx' instruction to become a 'mov eax' instruction, thus causing execution to resume from being_debugged.

That is a very subtle anti-debugging trick indeed.

Final note: the text of this paper was produced without reference to any *Microsoft* source code or personnel.

REFERENCES

[1] http://pferrie.tripod.com/papers/unpackers.pdf.

[2] Virus Bulletin, December 2008, p.4. http://www.virusbtn.com/pdf/magazine/2008/200812.pdf.

[3] Virus Bulletin, January 2009, p.4. http://www.virusbtn.com/pdf/magazine/2009/200901.pdf.

[4] Virus Bulletin, February 2009, p.4. http://www.virusbtn.com/pdf/magazine/2009/200902.pdf.

[5] Virus Bulletin, March 2009, p.4. http://www.virusbtn.com/pdf/magazine/2009/200903.pdf.

[6] Virus Bulletin, April 2009, p.4. http://www.virusbtn.com/pdf/magazine/2009/200904.pdf.

[7] Virus Bulletin, May 2009, p.4. http://www.virusbtn.com/pdf/magazine/2009/200905.pdf.

[8] Virus Bulletin, June 2009, p.4. http://www.virusbtn.com/pdf/magazine/2009/200906.pdf.

[9] Virus Bulletin, May 2010, p.4. http://www.virusbtn.com/pdf/magazine/2010/201005.pdf.

[10] Virus Bulletin, June 2010, p.4. http://www.virusbtn.com/pdf/magazine/2010/201006.pdf.

[11] Virus Bulletin, July 2010, p.7. http://www.virusbtn.com/pdf/magazine/2010/201007.pdf.

[12] Virus Bulletin, August 2010, p.4. http://www.virusbtn.com/pdf/magazine/2010/201008.pdf.

[13] Virus Bulletin, September 2010, p.12. http://www.virusbtn.com/pdf/magazine/2010/201009.pdf.

[14] Virus Bulletin, October 2010, p.16. http://www.virusbtn.com/pdf/magazine/2010/201010.pdf.

LETTERS

THE RELEVANCE OF SPAM FEEDS?

Building a good spam corpus is really important to validate ideas, to develop filters and to evaluate them. There is little literature available on the subject, so it was interesting to read the article 'On the relevance of spam feeds' in last month's issue of *Virus Bulletin* (see *VB*, October 2010, p.21), and to see that there are people working in this area.

However, I was a little disappointed with the article. In the introduction, the authors say: 'If the filters are not trained to detect a specific type of message, whether directly or indirectly, odds are that they won't detect any subsequent similar ones.'

The main idea behind some types of filters is to separate ham and spam messages. The main idea behind other types of filters is to detect and identify each kind of spam. It seems that the authors' filter belongs to the latter category. The two ideas lead to very different kinds of filters and to very different approaches to building a corpus of messages.

In the machine-learning community one efficiency parameter is the 'generalization ability'. This refers to the ability to learn from a small number of samples and classify newer ones which have never been seen before (this is explained in Vapnik's book and others).

This parameter depends heavily not only on the type of classifier but also on how the learning task is carried out and on the statistical characteristics of the incoming flow. There are usually a good number of messages to learn from – not too few and surely not too many.

Having a large number of messages is interesting, but is more useful for testing a classifier than for feeding the learning task.

In both situations (learning and testing), the spam feed should be representative of the real incoming flow.

The authors write about 'pollution', or errors, in the training corpus. The sensitivity of classifiers to errors in the training corpus varies. This depends not only on the kind of classifier but also (and mainly) on how they are trained. A number of papers have been written about this (D. Sculley, Gordon Cormack, Alexander Kolcz and John Graham-Cumming). For example, training methods known by the acronym TUNE (Train Until No Errors) will generate overfitted classifiers which are very sensitive to errors in the training corpus. It's good to remove errors, when they're found, but it's also interesting to evaluate the expected error rate.

Although I'm using vocabulary from 'statistical/machine learning'-based filters, the idea is valid for any kind of filter.

Next, the authors concentrate on the elimination of newsletters from the spam corpus. One should think a little more about this point. Should they really remove all newsletters from the spam feed or try to understand why there were a lot of newsletters in it? If the user of a spam feed is interested in understanding how newsletters (and 'grey mail' in general) are handled, they should remain there, and perhaps be labelled in a way in which they can easily be identified.

I was hoping to find an explanation of good coverage of the spam spectrum. The authors say something about it in the evaluation section, but this looks more like a recipe specific to their filters than scientific methodology for building a general purpose spam feed.

At one point the authors describe splitting the spam feed into a large number of clusters, each one related to a spam campaign. This is also something specific to their filter, and not useful in terms of building a general-purpose spam feed.

The authors conclude with the phrase: 'None of us can filter spam we do not receive ...' Again, this is a generalization of what they think a good filter should be. Maybe it should have read: 'Spam filters *like ours* can't filter spam we do not receive.'

In my opinion, if someone wants to create a good spam feed – which will be useful to many people – the best approach is to collect all spam messages arriving at a number of places, without any filtering, and without mixing them. In other words, the spam feed should be a statistical sample of all spam messages seen in a particular place. The spam feed should maintain the distribution of messages per genre (pharmacy, pornography, and so on). Users of the corpus will be able to adapt it to their needs: for training or testing filters, or for analysing spam traffic. In my humble opinion, this is the best way to build a spam feed that would fit the needs of everyone.

Jose Marcio Martins da Cruz, École des Mines de Paris

RESPONSE

An obvious difference in perception between the authors and Mr Martins da Cruz derives from the fact that he is an academic, while we are approaching the issue from a non-academic standpoint. As most readers know, this leads to very different goals and thus different reasoning. While the industry is interested in filters that have high accuracy and a short response time, most academics that I have met are looking for an elegant solution and a strong theoretical framework. Mr Martins da Cruz states that 'building a good spam corpus is really important to validate ideas, to develop filters and to evaluate them'. While that may have been the case a few years ago, when everyone was looking for solutions to filter spam, from the current perspective I disagree – a relevant spam feed is needed to filter spam.

The ideas may have been validated long ago, but spam is continually changing and spam tracking must keep up.

In the article we presented a method for evaluating a spam feed given the context created by other previously acquired spam feeds – to compute their overlap. It's an advantage to be able to measure the impact of a new feed prior to using it for a long period (and thus also usually paying for it). The questions we were trying to answer in this paper were 'How interesting is this spam feed for me?' and 'How can I make it better?'.

As Mr Martins da Cruz observes, we do assume that a message clustering method is in place (which is generally the case), but this does not mean that the clustering method is also a filtering method. The idea of clustering messages can conceptually be separated from the task of filtering ham from spam. The aim is to generate a good feed that can serve any type of filtering.

Although the direct usage of such a method becomes obvious when trying to buy a spam feed, it is also a method that is used to reduce the number of messages processed while losing the fewest possible representative concepts. The processed spam volume may not be a concern when tests are conducted on several thousand messages, but when spam feeds to be processed are measured in their millions per day, the concern is a major one.

The issues raised regarding last month's article refer to the processing of that feed, which is viewed as only interesting from a partitional clustering point of view, but moot from a hierarchical one. It is stated that 'In both situations (learning and testing), the spam feed should be representative of the real incoming flow.' While we agree with that point of view, we strongly disagree with the statement 'Having a large number of messages is interesting, but is more useful for testing a classifier than for feeding the learning task.'

Generalization ability for most clustering techniques (hierarchical AND partitional) is dependent on the representativeness of the concepts with which the system is trained. However, once the filtering method is chosen, the greater the number of different concepts the classifier is trained with, the higher the probability of obtaining a good cluster set at the end. A good initial overview is always preferable to having to infer from limited traces of data.

But the volume of spam is overwhelming for most machine-learning tasks. In attempting to reduce the sheer volume of samples that must be filtered, there is a significant risk that some existing spam types will no longer be represented. This is the undesirable outcome we're trying to limit.

From a machine-learning point of view, one might argue that industrial spam filtering now is even less exciting and challenging than it was five years ago. Simpler filters seem to work, whereas more sensitive, complex ones are being forgotten. This statement is, of course, subject to challenges from our peers. But a simple look at *Spamhaus*'s performance in the latest VBSpam tests (see *VB*, September 2010, p.22) shows the resilience of extremely simple filters that possess the capacity to process huge input volumes. The number of articles published and talks presented about spam filters over the last few years is also a good indication that many have given up on finding machine-learning techniques that have 99.5% accuracy and zero false positives. Bayes filtering doesn't work any more – at least not at the level that is required to pass tests. Although at some point in the past it did show promising results under laboratory conditions, given a small number of samples, it now fails the reality test.

Moving on to the issue of whether or not newsletters should be left within the spam feed, we must underline the fact that, depending on the way the spam feed is gathered, legitimate messages may be coming in along with spam, and that is a known nuisance. We do understand why newsletters are being mixed up with spam – as already argued in the article, one must emulate a real user in order to receive high-quality spam. But the problem with using those feeds in their initial form is exactly like using an annotated corpus for a static test when you know the annotators have mislabelled 5% of the items in there. This induces a level of uncertainty that is unwarranted, and any reduction of the percentage of misclassifications is welcome.

We thank Mr Martins da Cruz for his feedback.

Claudiu Musat, BitDefender

IS CYBERCRIME A BIGGER MONEY EARNER THAN DRUGS?

In the latest editorial (see *VB*, October 2010, p.2) you mention the rumour of cybercrime being bigger than the drugs trade ('...today, the profits generated by cybercrime worldwide are rumoured to match the revenues yielded by the illegal drugs trade'). Unfortunately this is something that keeps being repeated by people (and so some are beginning to believe it), but it's actually utter nonsense.

For more details see: http://www.theregister.co.uk/2009/03/27/cybercrime_mythbusters/.

Graham Cluley, Sophos

MEA CULPA

Thank you Graham.

Ed

END NOTES & NEWS

Black Hat Abu Dhabi takes place 8–11 November 2010 in Abu Dhabi, United Arab Emirates. For more information see http://www.blackhat.com/.

Infosecurity Russia takes place 17–19 November 2010 in Moscow, Russia. See http://www.infosecurityrussia.ru/.

AVAR 2010 will be held 17–19 November 2010 in Nusa Dua, Bali, Indonesia. See http://www.aavar.org/avar2010/.

The VB 'Securing Your Organization in the Age of Cybercrime' Seminar takes place 25 November 2010 in London, UK. 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 programme details and to book online see http://www.virusbtn.com/seminar/.

The 26th Annual Computer Security Applications Conference will take place 6–10 December 2010 in Austin, TX, USA. See http://www.acsac.org/2010/.

The 27th Chaos Communications Congress (27C3) takes place 27 to 30 December 2010 in Berlin, Germany. The Congress offers lectures and workshops on a multitude of topics and attracts a diverse audience of hackers, scientists, artists and utopians from around the world. For more information see http://events.ccc.de/.

Black Hat DC takes place 16–19 January 2011 in Arlington, VA, USA. For details see http://www.blackhat.com/.

The 10th Ibero-American Seminar on Information Technology Security will be held 7–11 February 2011 in Havana, Cuba. For details see http://www.informaticahabana.cu/en/home.

RSA Conference 2011 will be held 14–18 February in San Francisco, CA, USA. Early bird registration rates apply until 19 November 2010. For more details see http://www.rsaconference.com/2011/usa/.

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

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

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

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 Antimalware techniques: myths, reality and context'. A call for papers has been issued, with deadlines for submissions of 19 December for peer-reviewed papers and 12 December for non-reviewed papers. 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/.

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. For details see http://www.blackhat.com/.

VB2011 will take place 5–7 October 2011 in Barcelona, Spain. A call for papers will be issued in December 2010. More details will be available soon at http://www.virusbtn.com/conference/vb2011/. For sponsorship or any other queries, contact conference@virusbtn.com.

VB2012 will take place 26–28 September 2012 in Dallas, TX, USA. More details will be revealed in due course at http://www.virusbtn.com/conference/vb2012/. In the meantime, please address any queries to 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/