AUGUST 2005

virus BULLETIN

The International Publication on Computer Virus Prevention, Recognition and Removal

CONTENTS

2 **COMMENT**
The future's bright for (ex-)virus writers

3 **NEWS**
VB to take to the streets of Dublin
Spyware defined
Exclusion zone
Microsoft set to buy FrontBridge
Hoax alert
Sun, sea, sand and scams

4 **VIRUS PREVALENCE TABLE**

4 **TECHNICAL FEATURE**
Code emulation in network intrusion detection/prevention systems

7 **FEATURE**
Evolution from a honeypot to a distributed honey net

11 **Q & A REVISITED**
Deconstructing Windows Mobile

14 **COMPARATIVE REVIEW**
NetWare 6.5

20 **END NOTES & NEWS**

ISSN 0956-9979

IN THIS ISSUE

A BRIGHT FUTURE

Sasser author Sven Jaschan walked away free from a German court last month after receiving a 21-month suspended sentence for his crimes. With a job offer already in the bag, his future could be rosy. Costin Raiu looks at the fate of Jaschan and other virus writers and hackers who have been convicted.

page 2

A DECONSTRUCTION

Not satisfied by the answers provided by a *Microsoft* representative in last month's interview about security issues surrounding the *Windows Mobile* platform, Michael Moser takes matters into his own hands and delves a little deeper.

page 11

A NETWARE COMPARATIVE

Matt Ham finds *NetWare 6.5* to be significantly more tolerable than previous versions of the operating system. Find out whether the products for *NetWare* show similar improvement.

page 14

vbSpam supplement

This month: anti-spam news & events and Heather Goudey looks at methods of exploiting vulnerabilities in the human interface.

COMMENT

'[Jaschan] could be in high demand for teaching people how to protect themselves against viruses.'

Costin Raiu
Kaspersky Lab

THE FUTURE'S BRIGHT FOR (EX-)VIRUS WRITERS

There has been a lot of publicity lately about the author of the Sasser worm, Sven Jaschan, following his conviction by a German court. This got me thinking about other virus writers and hackers who have been caught, and the similarities between them.

What are the similarities between Sven Jaschan and, for instance, Robert Morris Jr? Well, they both wrote highly successful network worms, which caused chaos when they took over the Internet. They both confessed to writing their worms, although Sven Jaschan first had to be found by the police, who did this thanks to information provided by a couple of his friends (or ex-friends – I'm not sure whether Sven will be playing with them for a while, given that they sold him for *Microsoft*'s $250,000 bounty). Finally, both Jaschan and Morris received fitting sentences, which in Robert Morris's case put him back on track and, as far as I can tell, drove malicious software writing out of his mind.

What about the similarities between Sven Jaschan and Kevin Mitnick? Actually, in this case, there are more differences than similarities: Mitnick was a hacker and wasn't really into writing malware. Moreover, Mitnick didn't even write his own tools; he relied on ones created by his friend 'jsz' at Ben Gurion University in Israel. Sven Jaschan wrote his own viruses, carefully choosing the exploits which would allow his creations to spread at maximum speed on the Inter-*Windows*-net. Whereas Mitnick was driven by profit and the need to make a living, Sven Jaschan was a young man coding worms in order to show the world how clever he was. So there is no real similarity between Mitnick's sociopathic behaviour and Jaschan's utter foolishness.

It would seem that Sven Jaschan's profile is a lot closer to Robert Morris Jr's than Kevin Mitnick's. But this may not be the whole story.

Almost immediately after being apprehended, Sven Jaschan was hired by a security company, which promised to teach him to become a security programmer. With a suspended prison sentence of 21 months, and 30 hours of community service to pay for his crime, Jaschan can return to his studies knowing that he has a job in the security field, and a very bright future ahead of him (if he stays away from the malware game). Here, Kevin Mitnick's story is more than relevant. The author of two popular (they could even be called best-selling) security books, Mitnick is now a regular speaker at various security events, most notably the IDG Roadshows. This ex-hacker is now teaching people to secure their systems against the very things he was doing a couple of years ago.

I wouldn't be surprised if, in a couple of years, the reformed Sven Jaschan becomes the author of a couple of books (best-sellers, of course) on computer viruses and ways to attack networks. Thanks to his notoriety as the author of Sasser, he could be in high demand for teaching people how to protect themselves against viruses. The fact that these would be the very same people he infected with his creation years ago would be almost irrelevant. Sven Jaschan would have been turned overnight into a marketing symbol, which is exactly what happened with Mitnick.

Of course, Jaschan could write another worm, spread it over the Internet and go to jail, this time for good. Given the alternative, I think the choice is obvious.

The question is: who is at fault in the current situation, when reformed hackers and (why not?) virus writers, are talking about security, writing books which become best-sellers and being transformed into idols for the 12-year-old slashdot trekkie who has too much spare time? Is it the people who promote them, or the audience attending the seminars and buying the books? I'm afraid that the answer has serious implications. Moreover, I'm afraid that this issue will come back to haunt us in years to come. One question you should all consider: if Sven Jaschan writes a book, will you be buying it?

Editor: Helen Martin

Technical Consultant: Matt Ham

Technical Editor: Morton Swimmer

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

NEWS

VB TO TAKE TO THE STREETS OF DUBLIN

'A good puzzle would be to cross Dublin without passing a pub' – so said the character Leopold Bloom in James Joyce's *Ulysses*. Rather than taking up this challenge, however, *Virus Bulletin* is offering VB2005 delegates the opportunity to discover Dublin and its traditional music through a guided walking tour of famous pubs and bars in the city's Temple Bar area. The tour – which will take place after the close of the conference on the evening of Friday 7 October – will be led by musicians who will perform traditional tunes and songs and tell the story of Irish music as they lead delegates around famous Dublin pubs and bars. Places on the tour are limited and *must be booked in advance* – details of how to book are included in the VB2005 delegate pack, so register for the conference now to be sure of your place. For the full conference programme and online registration see http://www.virusbtn.com/.

SPYWARE DEFINED

The Anti-Spyware Coalition has proposed a solution to the tricky question of how spyware should be defined, and is inviting public comment on its proposal.

The Coalition, which is made up of industry members including *AOL*, *EarthLink*, *Yahoo!*, *Computer Associates*, *McAfee*, *Trend Micro* and *Microsoft*, describes spyware (and other potentially unwanted technologies) as: 'those that impair users' control over material changes that affect their user experience, privacy, or system security; use of their system resources, including what programs are installed on their computers; or collection, use, and distribution of their personal or otherwise sensitive information.' The full document containing the Coalition's proposals can be downloaded from http://www.antispywarecoalition.org/. The deadline for receipt of public comment is 12 August 2005, following which the ASC will respond to comments and produce a final document later in the year.

EXCLUSION ZONE

In a fit of pique, the developers of the free *Windows* EXE file encryption tool *PolyCrypt PE* have released a new licence agreement containing a set of very specific 'special conditions'. The new conditions exclude from the free licensing of the software all employees of *Kaspersky Lab* and any person or company associated with it. The developers' beef with the AV company is over its 'defamatory and libellous' labelling of *PolyCrypt PE* as a Trojan. All previous versions of the licence agreement have been revoked, and the developers warn that they will do the same for any other AV vendors that label their software as malicious. For the full licence agreement see http://www.interteq.net/.

MICROSOFT SET TO BUY FRONTBRIDGE

Microsoft announced its intention last month to purchase privately held email security firm *FrontBridge Technologies Inc.* for an undisclosed sum. The acquisition of *FrontBridge*, which provides a subscription service that incorporates email and instant messaging scanning, as well as compliance and archiving services, will be *Microsoft*'s second AV-related purchase of the year. The company completed its acquisition of security firm *Sybari Software* in June.

With *Microsoft*'s consumer security product *OneCare* currently undergoing beta testing, the purchase of both *Sybari* and *FrontBridge* is indicative of *Microsoft*'s plans to make a serious move into the corporate security market. The acquisition is expected to be complete by the end of September 2005.

HOAX ALERT

It has been a long while since *VB* reported on any virus hoaxes, but last month saw a new hoax email come to light after the launch in the UK of a campaign involving personal emergency contact numbers. The campaign, which gained momentum following the London bombings of 7 July, encourages mobile phone owners to add an entry in their mobile phone contacts list entitled 'ICE' (standing for 'In Case of Emergency'), and to enter against it the number of the person they would want to be contacted in case of an emergency. Should the worst then happen, the emergency services would quickly be able to find and contact the person's next of kin. Unfortunately the campaign is in danger of grinding to a halt thanks to the actions of hoaxers, who have taken it upon themselves to start a chain letter email urging people *not* to add an ICE entry to their mobile phones because, the email claims, there is a virus that will exploit it. Of course there is no such virus and the hoax achieves nothing more than to cause confusion and damage the campaign. Details of the ICE campaign can be found at http://www.icecontact.com/.

SUN, SEA, SAND AND SCAMS

Authorities in Malaga, Spain, must be congratulated on a bumper crop of arrests in connection with 419 scamming. A total of 310 people were arrested in an operation involving the FBI, the US Postal Service and the Spanish police. 'Operation Nile', which began in 2003, centred around a €100 million bogus lottery scam run by gangs operating from Southern Spain. Alongside the arrests, 166 properties were raided, with 2,000 cell phones, 327 computers, 165 fax machines, and €218,000 in cash being seized. Officials believe that the scam has claimed over 20,000 victims in 45 countries.

Prevalence Table – June 2005

Virus	Type	Incidents	Reports
Win32/Netsky	File	21,023	40.82%
Win32/Mytob	File	18,853	36.61%
Win32/Mydoom	File	3,755	7.29%
Win32/Bagle	File	1,928	3.74%
Win32/Bagz	File	1,594	3.10%
Win32/Agobot	File	691	1.34%
Win32/Lovgate	File	534	1.04%
Win32/Zafi	File	408	0.79%
Win32/Klez	File	325	0.63%
Win32/Mabutu	File	256	0.50%
Win32/Funlove	File	219	0.43%
Win32/Bugbear	File	170	0.33%
Win32/Dumaru	File	160	0.31%
Win32/Swen	File	131	0.25%
Win32/Pate	File	124	0.24%
Win32/Mimail	File	117	0.23%
Win32/Valla	File	106	0.21%
Win32/MyWife	File	102	0.20%
Win32/Sobig	File	101	0.20%
Win32/Fizzer	File	92	0.18%
Win32/Mota	File	71	0.14%
Redlof	Script	69	0.13%
Win32/Wurmark	File	60	0.12%
Win32/Sober	File	57	0.11%
Win32/SirCam	File	56	0.11%
Win32/Yaha	File	47	0.09%
Win32/Maslan	File	32	0.06%
Win32/Randex	File	25	0.05%
Win95/Tenrobot	File	25	0.05%
Win32/Gibe	File	24	0.05%
Win32/Eyeveg	File	22	0.04%
Win32/BadTrans	File	21	0.04%
Others[1]		304	0.59%
Total		51,502	100%

[1]The Prevalence Table includes a total of 304 reports across 59 further viruses. Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

TECHNICAL FEATURE

CODE EMULATION IN NETWORK INTRUSION DETECTION/ PREVENTION SYSTEMS

Aleksander Czarnowski
AVET Information and Network Security, Poland

If we look at our IT infrastructure from a security perspective it quickly becomes apparent that securing it will not be possible unless a thorough approach is taken. One of the best formal methods is risk analysis as it allows proper allocation of budget and placement of safeguards. However, one thing that is often overlooked (besides the fact that conducting a proper risk assessment is not a trivial thing and that it's hard to get it right the first time) is the need for two kinds of safeguard that should be placed within the business process. While currently we have a lot of safeguards that (try to) protect us from well-known, well-defined threats or vulnerabilities, the list of safeguards that can protect us from future attacks is not very long. One class of tools that tries to tie both ends together is Intrusion Detection and Prevention Systems (IDP/IPS). In this article we will look into only one problem in detecting attacks for new vulnerabilities: code emulation.

HOW TO DETECT THE UNKNOWN

To be able to detect not only known attacks at known vulnerabilities, but also new attacks targeting known vulnerabilities and new attacks targeting new vulnerabilities, we need to use a certain approach. Not every approach is applicable here, especially if it is intended for use in a real-life environment, but we can achieve the above partially by using one (or better yet all) of the following methods:

1. Identify some characteristic parts for a particular set of attacks and search input data for its occurrence.
2. Build an abstract model for a particular set of vulnerabilities and try to detect attacks based on this model by analysing input data.
3. Build an abstract model of results of a particular set of attacks and/or vulnerabilities and, based on this model, try to identify attack by analysing input data.

Note that not all of the methods listed above work well at network level or host level. Method 1 seems to be the easiest to implement, especially at network level, assuming we have a packet reassembly component in place. For example, many buffer overflow exploits use so-called 'nop-sled', which in its most trivial form is a stream of no-operation instructions. For example, the IA32 architecture has a single-byte NOP instruction that can be used. In fact, many exploits for IA32 systems use nop-sled based on a set of NOP instructions. Using this observation we can create a simple rule for detecting such exploits. In this case we don't need to know the particular exploit or the vulnerability being targeted by it at the time of creating the rule.

```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET
any (msg:"SHELLCODE x86 NOOP"; content:"|90 90 90 90
90 90 90 90 90 90 90 90 90 90|"; depth:128;
reference:arachnids,181; classtype:shellcode-detect;
sid:648; rev:7;)
```

Figure 1: Snort rule (from shellcod.rules file) detecting the simplest nop-sled for IA32.

TIME TO PREVENT?

Today's customers are not interested in detecting attacks – they simply want to keep the bad guys off their systems. To do this we need some kind of prevention measures. In our last case this seems to be simple. If we detect nop-sled, we can break the connection before it reaches the target. However, to deploy any mechanism in a real-life business environment we need a very low number of false positives and false negatives, otherwise it could impact the business more than the attacks themselves.

If you run snort or any other NIDS/IPS with a similar rule to that shown in Figure 1 you will quickly find out that the number of false positives can be high. Also, even if we need to use only single-byte instructions in nop-sled there are a lot more instructions we can use besides just NOP. Within the IA32 architecture we can use: cli, sti, cld, std, push, pop, popad, pushad etc. If we can control the target or predict how the EIP register will behave during overflow we can also use other instructions that occupy more than one byte in memory. Taking this idea further we can even use conditional and unconditional jump instructions. Figure 2 illustrates such a construction, while Figure 3 shows an IA32 example implementation.

When analysing Figure 3, take a look at the loc_40101E location. To be able to predict whether the jump will occur we need not only to analyse the code flow but also to keep the CPU state including registers and flags.

We also need to remember about the need to emulate unconditional jumps and calls. The code in Figure 3 is built

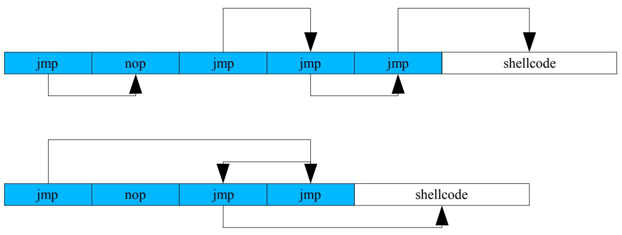


Figure 2: Different jump trampolines examples in nop-sled.

in such a way that, between jumps (that would occur on a real CPU) there is junk code which would influence the emulation process. We need to skip the junk code just as the real CPU does. We also need to remember about loop constructs. It may be possible to use an instruction like LOOP or JCXZ / JECXZ, which should also be emulated. Actually many polymorphic decryption loops use the LOOP or Jnn instruction on IA32.

Loops based on the LOOP instruction on IA32 are an interesting case because, at first, it would seem logical that this is how a compiler would generate the following C code:

```
int i;
for(i = 0;i < 10;i++)
        __asm__("nop");
```

However, both GCC and VC compiler use conditional jumps. Here is a disassembly of the above loop compiled with GCC 2.95.3 on OpenBSD 3.1:

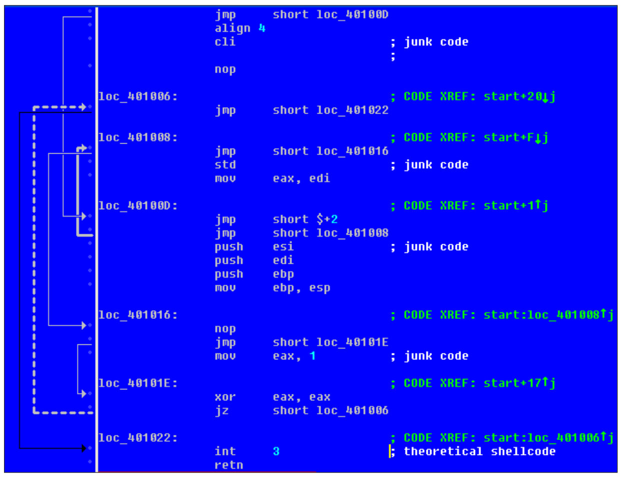
```
0x17a3 <main+11>:  movl   $0x0,0xfffffffc(%ebp)
0x17aa <main+18>:  nop
0x17ab <main+19>:  nop
0x17ac <main+20>:  cmpl   $0x9,0xfffffffc(%ebp)
0x17b0 <main+24>:  jle    0x17b4 <main+28>
0x17b2 <main+26>:  jmp    0x17bc <main+36>
0x17b4 <main+28>:  nop
0x17b5 <main+29>:  incl   0xfffffffc(%ebp)
0x17b8 <main+32>:  jmp    0x17ac <main+20>
0x17ba <main+34>:  nop
0x17bb <main+35>:  nop
0x17bc <main+36>:  xor    %eax,%eax
0x17be <main+38>:  jmp    0x17c0 <main+40>
0x17c0 <main+40>:  leave
0x17c1 <main+41>:  ret
```

If we remove the nops and function epilog we end up with the following construction:

```
0x17ac <main+20>:  cmpl  $0x9,0xfffffffc(%ebp)
0x17b0 <main+24>:  jle   0x17b4 <main+28> ;if i <= 9
                                          ;goto 0x17b4
0x17b2 <main+26>:  jmp   0x17bc <main+36> ;exit from
                                          ;loop
0x17b4 <main+28>:  nop                ;execute loop scope
0x17b5 <main+29>:  incl  0xfffffffc(%ebp) ;increment
                                          ;loop counter
0x17b8 <main+32>:  jmp   0x17ac <main+20> ;jump to the
                                      ;beginning of loop
0x17bc <main+36>:  xor   %eax,%eax    ;executed after
                                      ;loop ends
```

This observation allows us to make a rule to look for loop constructions that are not typical for compiler-generated code. This could be done with our emulator. However we still have to keep in mind that some applications besides shellcodes are written in assembly language. It is also important not to take our assumption about shellcode construction too far. For example, the assumption that every shellcode is written directly in assembly language is false.

Some of the first publications about buffer overflow exploitation on Win32 platforms were based on writing shellcode in C and doing cut-and-paste operations from a high-level language compiled disassembly. So we can find shellcode which looks just like compiler-generated code.



```
                jmp     short loc_40100D
                align 4
                cli                         ; junk code
                                            ;
                nop

loc_401006:                                 ; CODE XREF: start+20↓j
                jmp     short loc_401022

loc_401008:                                 ; CODE XREF: start+F↓j
                jmp     short loc_401016
                std                         ; junk code
                mov     eax, edi

loc_40100D:                                 ; CODE XREF: start+1↑j
                jmp     short $+2
                jmp     short loc_401008
                push    esi                 ; junk code
                push    edi
                push    ebp
                mov     ebp, esp

loc_401016:                                 ; CODE XREF: start:loc_401008↑j
                nop
                jmp     short loc_40101E
                mov     eax, 1              ; junk code

loc_40101E:                                 ; CODE XREF: start+17↑j
                xor     eax, eax
                jz      short loc_401006

loc_401022:                                 ; CODE XREF: start:loc_401006↑j
                int     3                   ; theoretical shellcode
                retn
```

Figure 3: Theoretical jump trampoline – flow control analysis done by IDA Pro based on static disassembly.

CODE EMULATION IN PRACTICE

Now let us assume that we have built a code emulation engine that can read payload from incoming packets and try to execute it in a sandbox to evaluate whether it is legal or dangerous code that should be stopped. As one might expect, there are many problems with such an approach. For now, we will ignore the use of polymorphic shellcode in the wild. First we will try to define the requirements for such an emulator.

To be more than just a signature-based system we need to keep the state of the emulated CPU. This raises an important question: which CPU should we emulate? A better question would be how to distinguish from the packet payload which architecture those opcodes are for. One very crude solution would be to map the destination IP addresses with a particular system platform and CPU architecture. Knowing the operating system of the target could also be useful because it could help the emulation process. For example, we can ignore, to some extent, the system calls in *Linux* shellcode (int 80h) if the destination IP is a *Windows* system (system calls are made with the help of the call instruction). Now we see another issue: we need to emulate system calls, or at least be able know what particular functions do to evaluate code. While in *Linux*/Unix systems this could be easy, it could be a bit trickier for *Windows* or IOS systems, where calls or jumps are being made to execute system functions.

Imagine for a moment that we have solved all of the above problems. After all, we could do some statistical tests to determine CPU type, or try to disassemble part of the payload to see if we can get something meaningful for the particular architecture. We don't need to emulate all of the called functions – we can look only for those which are typical for shellcodes. At this point we come across another important problem: is it data or code?

This is a very important question. In the case of IA32 there is no easy way of telling whether the particular byte stream is data or code. So if we emulate only the payload from one packet we don't know whether it is real code or just data. This is why some snort shellcode rules are disabled immediately, because of the high number of false positives. Look at Figure 1 again – theoretically, a perfectly legal application can contain such a byte stream in its data section.

One solution would be to reassemble the packet stream, extract all payloads and put it into our emulator. Unfortunately we still don't know where the data and code lie in memory. So the above solution could be applied in the case of sending an executable object of known format like ELF, A.OUT or a PE file. However, it would fail in the case of shellcode.

Another approach would be to analyse traffic at application level. In some cases it would be easier to distinguish data from code. This approach requires the IDS/IPS system to be aware of the particular server application and interact with it. There are very few solutions that can do this currently for more than one application. Usually application firewalls/ IDS/IPSs are designed to work with one particular application like *IIS*.

We have ignored polymorphic shellcode (Figure 4) until this point. The main idea behind it is to make signature-based detection impossible. To achieve this, real shellcode code is encrypted and encapsulated into a new structure that contains nop-sled built with many different instructions and a decryption loop. After decryption the execution flow is directed at real (decrypted at this time) shellcode.

n0P SLE4d	Decryption loop	Encrypted shellcode	Ret address

Figure 4: Polymorphic shellcode.

Even a very simple INC/DEC or XOR encryption loop hides strings like /bin/bash or cmd.exe and system function calls which are typical for shellcode operation. If the new nop-sled is built from many different instructions that can occupy more than one byte and the decryption loop is random, a simple signature-based system will not be able to detect it without a high level of false positives. Some solutions like [1] use a characteristic-based approach looking for particular bytes. Another approach has been suggested in [2]. Here the authors propose a method based on analysing the return address of shellcode. If it lies within a particular range we can assume that it is a working exploit. The one problem is that we still need to be able to identify the area of ret address space within the packet's payload.

Emulating even a simple decryption loop takes some CPU resources. Decrypting encrypted shellcode can be easy if there is a decryption key or the key is a byte or word value. But what will happen if the decryption loop uses a brute force attack to decrypt the shellcode because there is no decryption key included? This method has already been used in malware [3] and there is no reason why attackers couldn't use it at network level. As one might guess this poses a serious problem to a code emulator: it would either fail or use a lot of system resources. It would also take time to evaluate code, which brings us back to the business requirements that must be met to deploy a solution in a real-life environment.

SUMMARY

The aim of this article was to scratch the surface of the real problem that IDS/IPS vendors try to battle. Personally, I think they are losing this battle for now. It seems that some attack detection should be done strictly at host level. HIDS should also introduce additional safeguards at the system level to stop particular attack classes. Such architecture should be strengthened with network-based IPSs. It will be interesting to see for how long we will have to deal with buffer overflow vulnerabilities. The introduction of different stack protection techniques in conjunction with safe versions of C/C++ functions [4, 5] should make buffer overflow attacks extinct within the next five to eight years. Then the presented emulator will have many more problems to battle.

BIBLIOGRAPHY

[1] 'Polymorphic Shellcodes vs. Application IDSs', Next Generation Security Technologies, http://www.ngsec.com/docs/whitepapers/polymorphic_shellcodes_vs_app_IDSs.PDF.

[2] Archana Pasupulati, Karl Levitt, S. Felix Wu, 'Buttercup: Network-based Detection of Polymorphic Buffer Overflow Exploits', Department of Computer Science University of California, Davis, http://www.citris.berkeley.edu/events/spotlight/posters/posters_1003/1_felix_wu.ppt.

[3] Peter Ször, 'Bad IDEA', *Virus Bulletin*, April 1998, http://www.peterszor.com/idea.pdf.

[4] StrSafe http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsecure/html/strsafe.asp.

[5] SafeStr http://www.zork.org/safestr/.

FEATURE

EVOLUTION FROM A HONEYPOT TO A DISTRIBUTED HONEY NET

Oliver Auerbach
H+BEDV, Germany

Over the last few years worms and bots in particular have become a penetrating widespread threat. Anti-virus companies have developed better and better heuristic detection against these pests, but some are still slipping through. Therefore, the old method of adding signatures as soon as a new variant shows up remains very important. At a time when more than 40 new bot variants are appearing each day, it is extremely important to have a binary sample of each for analysis in house.

A traditional honeypot that captures the latest variants is quite an efficient technique. However, a slight disadvantage of the technique is that most of the attacks will be from the same subnet and after a while you will be aware of all the bots around the honeypot, and fewer new variants will be discovered.

For increased efficiency, more and more honeypots must be set up in different locations, especially in different subnets. Usually this requires a large amount of administrative effort, involving the fine-tuning of each of the honeypots' behaviour each time a new infection technique or exploit is discovered.

This article describes how we managed to extend our honeypot, which was specially designed to capture worms, to a honey net that solves this problem using a new technique. All information about new attacks and samples are collected at a single point. Further configuration and changes can be made from a single point and all of this is possible in real time.

AN IDEA WAS BORN ...

One Sunday evening in 2004, I spent some time watching connection attempts on various ports (such as 135, 139 and 445) on my router at home and was amazed by how many were blocked. From previous analysis in our virus lab, I knew that these were attempts to infect my machine with copies of worms like Lovsan, Sasser, Korgo and various other bots.

In order to capture a binary I redirected one of those ports to a *VMware Windows XP* machine without SP and without patches. After a couple of minutes, I managed to get a binary copy of a bot that seemed to be a brand new variant since it was not detected by our product at that time. The idea of an automated system was born.

RESEARCH

Only a couple of months earlier, I had played with the fingerprinting tool called *nmap* [1]. I saw it as a requirement that our honeypot should behave in the same way as the vulnerable *Windows XP* machine mentioned above. If an attacker uses this tool or any other fingerprinting technique, he should come to the conclusion that this is a real machine waiting to be infected.

From time to time, I see port scans on several of the well-known ports without any infection attempt to follow. This could be some kind of automated or manual collection of IP addresses for an infection process planned to take place at a later stage. It could also be an attempt to determine whether the machine matches certain conditions.

Be that as it may, the machine should respond in the same way as a regular machine on most ports. I felt that it would be best to create a honeypot that is able to simulate more than just a machine including some standard services. Moreover, it should be possible to simulate a Mydoom backdoor or a vulnerability such as the one Opaserv uses, for instance.

Finally, I decided to use *WinpkFilter* [2], a packet-filtering framework that gives you full control of each packet arriving and leaving the machine. According to Lance Spitzner's classification of honeypots [3], I would categorize this as a low interaction honeypot, specially designed to capture the binaries of worms and nothing more than that.

BASIC IMPLEMENTATION

The first version was able to simulate the MS04-011 vulnerability [4], which was being used extensively by W32/Korgo at that time, and the famous DCOM RPC vulnerability MS03-026 [5], which was first used by W32/Lovsan (alias Blaster).

By simulating these two vulnerabilities, I was able to capture 11,190 working binaries within three weeks in August 2004. It was rather interesting that a significant number were infected with the file infector virus W32/Parite. These were various infected worms that merely carried the old file infector along with them. Statistics from the first three weeks after the honeypot went online can be seen in Figure 1.

```
2777   Worm/Korgo.U
1399   Worm/Korgo.S
605    Worm/Rbot.DO
554    Worm/Korgo.X
543    Worm/Rbot.JL
436    Worm/Rbot.DA
370    Worm/Rbot.GT
310    Worm/SdBot.JG
295    Worm/Korgo.Q
293    Worm/Korgo.P
288    W32/Parite
3320   Other
----------------------
11190 Total
```

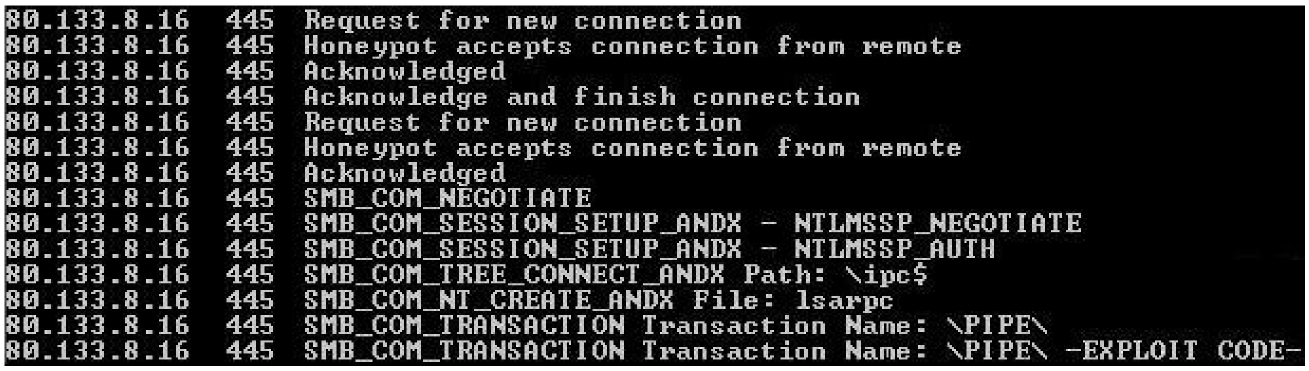
Figure 1: Honeypot statistics from a three-week period in August 2004.

Overall, 142 different variants of various worms were captured. After a very successful start, with sometimes as many as a dozen new variants a day, fewer and fewer new binaries were discovered. After a while, I started to implement more vulnerability behaviour as well as support for some common commands from the SMB protocol that were used by worms in order to propagate (see Figure 2).

After another peak of new variants, things returned to the previous low level of activity. An analysis of the source IP addresses showed that around 80 per cent of the infections originated from the same subnet. At that time, the honeypot was located in the *Deutsche Telekom AG* subnet and the IP address was 80.133.x.x.

Deutsche Telekom is the largest ISP in Germany and the connection with the Internet is terminated each day. Therefore, you will get a new IP address each time you dial up. Unfortunately, I found myself always in the 80.133.x.x range, even if I reconnected dozens of times. I was sure that there were other, yet-to-be-seen variants that were active in other subnets.

The reason is that most worms try addresses that are similar to their own, in order to increase their chances of success. Usually this trick is guaranteed to increase the chances of a



```
80.133.8.16   445  Request for new connection
80.133.8.16   445  Honeypot accepts connection from remote
80.133.8.16   445  Acknowledged
80.133.8.16   445  Acknowledge and finish connection
80.133.8.16   445  Request for new connection
80.133.8.16   445  Honeypot accepts connection from remote
80.133.8.16   445  Acknowledged
80.133.8.16   445  SMB_COM_NEGOTIATE
80.133.8.16   445  SMB_COM_SESSION_SETUP_ANDX - NTLMSSP_NEGOTIATE
80.133.8.16   445  SMB_COM_SESSION_SETUP_ANDX - NTLMSSP_AUTH
80.133.8.16   445  SMB_COM_TREE_CONNECT_ANDX Path: \ipc$
80.133.8.16   445  SMB_COM_NT_CREATE_ANDX File: lsarpc
80.133.8.16   445  SMB_COM_TRANSACTION Transaction Name: \PIPE\
80.133.8.16   445  SMB_COM_TRANSACTION Transaction Name: \PIPE\ -EXPLOIT CODE-
```

Figure 2: Support was implemented for some common commands from the SMB protocol that were used by worms in order to propagate.

worm's success both because similar addresses are more frequented than randomly generated ones, and because there is a high chance of address validity [6].

Eventually, my colleagues from the virus lab helped me out and set up new traps in other subnets. Even those who live just a few kilometres away might be in a different subnet, presupposing that their dialup receiver provided by the ISP is a different one.

The distribution of the traps worked pretty well and we got new variants of worms that we had not seen before. Every time a new trap was put online, we were able to see another peak.

A mailing routine was added to collect all the samples on a central server. The binary was shipped along with the original IP and port, timestamp, original filename and the whole infection log, which could help to determine bugs in the program as well as transmission errors or ideas for further implementation. At the time the email arrived on the server, a couple of error checks were made, a decision was made as to whether it should be forwarded to the lab for further analysis, and statistics were gathered.

The conclusion seems obvious: it is important to have as many traps as possible, keeping in mind that they should be in different subnets.

All this gave rise to other problems as I had to maintain more than one trap, update it, collect and interpret the results. Moreover, I had data only from successful infection processes. Imagine if there is a new technique to compromise a machine or there is the need to implement further commands from the SMB protocol or even listen on other ports.

From the current point of implementation, there were some drawbacks that forced a redesign of the whole concept and that practically made the change from one or more stand-alone honeypots into a distributed honey net.

EVOLUTION

The new recommendation was to design a system with as many traps as possible. Although it should be possible to collect and watch the infection processes at a single point, the solution was to split the honeypot into two parts. The first is called the 'Forwarder' and the other is called the 'MainPot'.

As you might have anticipated, the Forwarder will redirect the traffic to a certain IP address and a certain port that can be configured using a config.cfg file along with the installation process. The traffic arriving on certain ports at the Forwarder side is not just a simple redirect using NAT [7]. As we wanted to create reliable statistics, to watch out

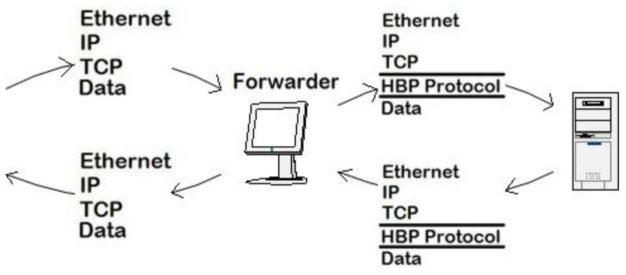


Figure 3: Another layer on top of the TCP protocol was implemented.

for seeding attempts of new malware and to figure out in which subnets certain malware is active, another layer on top of the TCP protocol was implemented (see Figure 3).

The new protocol (see Figure 4) is 16 bytes long and contains data such as an identifier, original IP, original source and destination port, number from the config.cfg file and some free bytes reserved for further ideas.

The identifier helps the MainPot to distinguish whether the packet really comes from a Forwarder, if it was just a port scan or any other data arriving on that port. The number set in the config.cfg file could be anything from 1 to 65,535 and uses two bytes in the protocol. This number must be set manually during the Forwarder's first installation process.

The purpose of this is so that each Forwarder can be recognized, even if the IP address changes. It makes sense to know which ISP and location you are using, as the ISP might implement port filtering on common malware ports from one day to another and you won't receive anything and might wonder why. Exactly this happened to one of the Forwarders placed in Bucharest, Romania in June 2005.

Other important data is the original source and destination port. It is necessary to ship this data along, as the MainPot must know how to react when constructing the answer packet. The original IP address is shipped in order to provide statistics.

However, the original IP address, original source and destination port are needed later when the Forwarder has to strip down the protocol and construct the final answer packet. In fact, this is the same thing the MainPot does in the first step when a new packet arrives. Using the original ports and addresses the packet is reconstructed as it arrives

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Identifier                  |    Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Reserved           |   Identifier from config.cfg  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Original Source Address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Original Source Port     |   Original Destination Port   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4.

on the Forwarder side. It is then saved to a log file including IP + TCP + Data – even the Ethernet layer is saved there. The purpose of the saved packets is described in the next section.

The MainPot searches its database for the type of data that the answer packet should contain. If it is successful, which means that this was already implemented, it constructs the answer packet. Finally, the HBPot protocol header is built in again and sent to the Forwarder. This has the simple task of replacing the original IP and the ports, then recalculating the checksums before sending the answer packet to the attacker.

Depending on the connection and location, the delay between the Forwarder and the MainPot is around 200ms. Each of the honeypot parts needs less than 10ms to construct the packet, which means that most of the time is spent 'on the road'.

COLLECTING DATA

As mentioned before, every incoming packet including all layers is saved to a file. It is not only the incoming packets but also the outgoing packets that are stored in this file.

The infection log is saved in a folder with the name of the original source IP address combined with the timestamp of the first communication. The time frame during which this is valid is only a couple of minutes – the reason being that (at least in Germany) your IP address is different every time you connect to the Internet. A new connection attempt at some later point will result in the creation of another folder, in order not to append the previous infection log with data that should be kept separately. Of course, this could be a reinfection attempt from the same machine, but it is also possible that it is a different machine that is using this IP address now.

Nevertheless, the infection log is very important when it comes to unsuccessful infections. There could be many reasons for these, such as a broken Internet connection, the machine on which the worm is running having been turned off during the transfer, and so on.

It is rather interesting when a new technique is used in order to infect a machine. This could be a new exploit or just another command from the SMB protocol that has not yet been implemented and it will simply result in the packet not being answered at the MainPot side.

Let's assume that the never-seen-before exploit code 'MSxx-xxx' is used and it is not yet implemented. The infection log would save the communication up until the point at which the MainPot doesn't know how to go further. In order to implement the next step, I have created a tool that is able to read the infection log and simulate the whole communication against the aforementioned vulnerable *Windows XP VMware* machine in a secure environment.

The whole communication is simulated, starting with the three-way handshake up to the last packet and finally I can figure out what the MainPot should have had answered in order to go further with the communication.

At this point, I am able to implement this answer package into the MainPot and hope for an attacker using the same exploit to go another step further. A binary – probably from another source – would help a lot as I could simply create a full infection trace using Ethereal and implement all necessary communication in a single step. Without a binary to analyse in the virus lab, this is the only way to go further and finally capture the binary by myself.

There are some ideas that the simulation against a vulnerable *VMware* machine could take place in real time as long as the communication with the Forwarder is pending. At the time of writing, this has not yet been implemented and I cannot say if this is fast enough as the connection is usually dropped within seconds if no answer packet has arrived.

FINAL CONSIDERATIONS

In order to create a successful honeypot it is important to have as many sensors as possible. In fact, it does not really matter if these are Forwarders or stand-alone honeypots, although the forwarder technique makes a lot of things easier and allows real-time changes and analysis for the whole honey net.

REFERENCES

[1] Nmap: http://www.insecure.org/nmap/.

[2] WinpkFilter: http://www.ntkernel.com/.

[3] Lance Spitzner, *Honeypots: Tracking Hackers*, Addison-Wesley, 2003, p.73.

[4] Microsoft Security Bulletin MS04-011: http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx.

[5] Microsoft Security Bulletin MS03-026: http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx.

[6] Gabor Szappanos, 'Advanced survival techniques in recent Internet worms' *Proc. Virus Bull. Int. Conf.*, 2004 (abstract: http://www.virusbtn.com/conference/vb2004/abstracts/gszappanos.xml).

[7] NAT: http://en.wikipedia.org/wiki/Network_Address_Translation.

Q & A REVISITED

DECONSTRUCTING WINDOWS MOBILE

Michael Moser
IBM Research GmbH, Switzerland

Last month's issue of *Virus Bulletin* contained an interview conducted by Juha Saarinen with *Microsoft* representative Brett Roberts about the update issues surrounding *Microsoft*'s *Mobile* operating system platforms (see *VB*, July 2005, p.13). However, Mr. Roberts' answers were left largely uncontested, so the following are a few points that I felt were missing from the article.

[*As a supplement to this article, VB's Technical Editor Morton Swimmer provides a brief overview of the Microsoft Windows Mobile platform on p.13 - Ed*]

NO ESCAPE

Juha's first question related to whether (unlike on the PC platform) it is safe to run a *Windows Mobile* device without updates. However, the answer provided by Mr. Roberts did not include a discussion of securing the operation of the device – for example, what is done or offered by the OS, so that a malevolent piece of software (such as one that always shuts off the device and/or reinstalls itself during reboot) cannot prevent the user from regaining full control of his/her device?

Windows Mobile (*WM*) has no 'restricted' or 'safe-mode' to allow rebooting while skipping the normal startup programs, thus allowing the recovery and cleanup of an infected system. For a common user the dreaded operation that in *Windows CE* parlance is known as 'hard-reset' – a complete memory re-initialization and reset to factory defaults (which also implies 100 per cent data loss) – appears to be the only available 'solution' to resolve such a situation [1].

Furthermore, there is no encapsulation of processes that would ensure limited threats from flawed drivers or other third-party software that comes pre-installed on the system. For example, some Bluetooth stacks are well known to have security holes. There is no OS support for executing certain parts of the software in a restricted or limited mode to prevent access to and/or modification of critical OS data, amongst other things.

Mr. Roberts emphasizes the rather trivial point that not getting infected is the best way to stay uninfected – i.e. do not download and/or install 'doubtful' software. This needs to be emphasized because, once the system has been infected, *Windows Mobile* does not have any further system-internal protection, access controls, or any other safety barriers. However, a program does not even have to be explicitly malevolent in order to cause disruption. Just a simple program error can be enough to wipe out the entire system and/or render it useless (by causing endless reboots, etc.).

NO, NOT NEVER

Further on in the Q & A, Mr. Roberts states that it is difficult, if not impossible, to distribute patches directly to end-customers due to the diverse nature of *Windows Mobile* devices. He claims that *Microsoft* depends on the hardware manufacturers and OEMs to integrate and forward its patches.

Alas, the hardware manufacturers and OEMs seldom feel the need to forward these patches with the due urgency. My own experience (with multiple generations of devices from *Casio*, *Toshiba*, *Siemens* and several *Compaq*/*HP iPAQ*s) is that the release of such updates takes at least two, but more typically three to six months, during which time users remain exposed and essentially can do nothing about it. The burden, therefore, is on the user to avoid being exposed to malicious software. This problem is revisited throughout the article.

However, I find this situation hard to accept, since many sources indicate that, when a manufacturer requests to license *Windows Mobile*, *Microsoft* has a pretty strong say in what a device must offer, how it must appear and behave, what applications must be included and which must *not* come pre-installed, etc.

Given such power, it would seem comparatively easy for *Microsoft* to enforce a standardized API, via some code package (of course sealed, signed, etc.) that would be able to reflash at least parts of the OS without requiring an OEM's further blessing and support.

Given that we have already gone through several generations of *Windows CE* / *Handheld PC* / *Pocket PC* / *Windows Mobile* and whatever-they-name-them-today devices, there would have been ample time to require such a capability and thus I rather suspect that the issue was not considered very important or urgent (to date). I wish the *Mobile* group had exhibited the same 'great work by the *Windows* division on *Windows* update technology'.

NO DISCRETIONARY EXECUTION OF CODE

Mr. Roberts points out that the *Windows Mobile* platform can also be operated as a closed platform (using techniques from the Trustworthy Computing Initiative), where operators or businesses have full control over what software can be installed to a device, thus minimizing the risks. This, of course, is just a way of sidestepping the real problem by merely preventing discretionary execution of code – and due to its restrictive nature this is not a popular or widespread approach.

Furthermore, operating as a closed platform to minimize risks is not 100 per cent secure, since (as previously indicated) the code does not even have to be malevolent to cause problems; a simple bug is enough!

A controlled environment simply reduces the probability of software bugs by making the assumption that approved applications are tested more thoroughly before they are widely deployed, especially regarding their interaction with other controlled applications. This is probably a faulty assumption and a dangerous one.

NO MONOCULTURE

Interestingly, while Mr. Roberts laments that the *Pocket PC* ecosystem lacks the uniformity that would allow simple and widespread distribution of security updates, he fails to mention that exactly such diversity is probably a major safeguard for these systems!

The other reason why we haven't seen any major attacks to *Windows Mobile* platforms yet is *Windows Mobile*'s lack of capabilities that could be abused by worms and viruses: none or only very few browser plug-ins, very limited scripting support, no powerful command line processor, built-in applications having no macro capabilities, etc. But the opportunities are growing, with each operating system and application generation becoming more powerful than the last.

NO CONNECTIVITY

Finally, today's PDAs are still far less continuously connected to the Internet than desktop systems, and in many cases they connect to the Internet only via some 'sync partner'.

As a result, there is only a relatively small population of attackable devices available on the Internet at any point in time, and the channels via which worms and viruses would have to travel to infect them are convoluted and contain barriers that need to be crossed. But, with more and more *Windows Mobile* devices being constantly and directly connected to the Internet (thanks to integrated WiFi and/or built-in high-speed phone capability, e.g. GSM/GPRS or UMTS, and especially with billing models based on generated traffic rather than connection time) this picture may change quickly and dramatically.

NO PLACE TO HIDE

There is one major risk on the horizon for *Windows Mobile*, that is quite troubling and that was not mentioned at all in Juha's article: most of the newer *Windows Mobile* devices have a large flash memory in which not only do they store the OS and the pre-installed applications, but a part of the memory (usually a quarter to a half) is also made available to the user as a place where he/she can install applications and store data. The feature was meant originally to provide users with a place where they could store data that would be safe even when the device runs out of battery and all normal RAM content is lost.

The flash memory on such devices is, in effect, split into two 'partitions'; one for the OS and pre-installed applications and one for user data. However, the boundary between these two partitions is purely virtual and can be shifted, thereby allowing a newer (and most likely larger) version of the OS to consume a slightly larger fraction of that memory than the previous one.

Such a mechanism also means that, given the device-specific or brand-specific knowledge on how to write data to the flash, viruses, too, could 'burn' themselves into the OS area of the 'ROM' such that they would survive hard-resets, i.e. complete 'memory loss' and reset to factory defaults. Such viruses would not be removable with any means available to the normal user, meaning that an infected device would have to be sent back to the manufacturer for repair.

That, together with the growing CPU [2, 3] and architectural homogeneity of *Windows Mobile* devices, will become a major threat in the probably not-so-distant future. The requirement for hardware-assisted memory protection and OS-supported security models will thus soon become a requirement for *Windows Mobile*-based PDAs and phones.

END NOTES

[1] There is now a third-party tool available that offers exactly such a feature (see http://www.monocube.com/), but the standard *Windows Mobile* lacks that capability.

[2] Earlier versions of *Windows CE* ran on different CPUs (MIPS, SH-3/4, misc. *Philips* processors, etc.), while versions since *Pocket PC 2002* run only on XScale and other StrongARM-compatible processors.

[3] For an overview of CPUs and architectures *Windows Mobile* runs see Chris de Herrera's excellent compilation at http://www.pocketpcfaq.com/version.htm.

THE MICROSOFT WINDOWS CE PLATFORM

Morton Swimmer
IBM Research GmbH and Virus Bulletin

Since there is a lot of confusion over what, exactly, encompasses the Microsoft Windows Mobile platform, VB's Technical Editor Morton Swimmer has provided the following brief run-down of the variations of the system - Ed.

The portable and embedded operating system platform that *Microsoft* offers is a very complex beast. There is the core operating system, *Windows CE*, which is currently at version 5.0, and then the application layer, which is also at 5.0. However, in between, the vendors using the products have mixed and matched as required. To make things just that much more interesting for us, the final product name has morphed from offering to offering and currently stands at *Windows Mobile*.

The following table is a list of product names with the corresponding operating system and application levels as well as the CPUs these offerings ran on. It is not definitive and was compiled to the best of anyone's knowledge.

Although the *Windows CE* Platform has supported many CPUs in the past, currently only the *Intel* Xscale and *Samsung* S3C2410 seem to be supported. However, within the .NET framework, the developer may compile to MSIL, which is a CPU-independent language allowing the application to run on any platform for which there is a cross compiler. Of course, this is an equal opportunity for malware writers to obtain wide code coverage despite the variations in the architectures of these devices. Only time will tell if this feature can effectively be abused.

Product Name	Year released	OS level	Application level	Known CPUs supported
Handheld PC 1.0	1996	1.0	1.0	SH3, Vr4101
Handheld PC 2.0	1997	2.0	2.0	SH3, Vr4101, PR31700
Palm-size PC 2.0	1998	2.01	1.0	n/a
Palm-size PC 2.11 Chinese Version	1998	2.11	1.1	n/a
Handheld PC, Professional Edition	1998	2.11	3.0	Vr4111, Vr4121, R4000, SH3, TX3912, Pr31700
Palm-size PC 2.11 US, Japanese Version	1999	2.11	1.2	Vr4111, Vr4121, R4000, SH3, TX3912, Pr31700
Pocket PC[1]	2000	3.0.9348	3.0	StrongARM, XScale, Vr4121, SH3, TC3922, Pr31700
Handheld PC 2000	2000	3.0	3.0	StrongARM, XScale
Pocket PC 2002[2]	2001	3.0.11171	3.0	StrongARM, XScale
Pocket PC 2002 Phone Edition	2002	3.0	3.0	StrongARM, XScale
Windows CE .NET 4.1	2002	4.1	4.1	n/a
Smartphone 2002	2002	3.0	3.0	StrongARM, XScale
Windows Mobile 2003[3]	2003	4.20.1081	4.2	Xscale, S3C2410
Smartphone 2003	2003	4.20.1088	4.2	Xscale, S3C2410
Windows Mobile 2003 Second Edition[4]	2004	4.21.1088	4.2.1	Xscale, S3C2410
Windows Mobile 5.0	previewed 2005	5.1.1700	5.0	n/a

[1] StrongARM and Xscale are largely compatible.

[2] There were three service packs for the Pocket PC 2002 edition. Not all were offered by the vendors.

[3] Supports .NET.

[4] First malware reported for this platform.

COMPARATIVE REVIEW

NETWARE 6.5

Matt Ham

Those who have read *Virus Bulletin*'s previous reviews of *NetWare* products will be familiar with my views about the platform – overall, I have found the platform less than convenient to work with and the products themselves generally even worse.

To be reasonable, however, *NetWare* has become significantly more tolerable with version 6 and newer, though to a certain degree this is a function of the fact that hardware has only recently been able to deal with the demands of *NetWare*'s GUI. Thankfully, the GUI in *NetWare 6.5* has been relieved of the images of eccentric gymnasts which graced version 5, which has also made the review process a little more bearable.

With the improvements to the operating system, therefore, it was left to the products to determine whether the review experience would be pleasant or otherwise. One issue made itself known early on: several products caused message boxes to pop up on the client when viruses were detected on the server, and there was no obvious way to remove this feature. With large test sets the added network traffic slowed down scanning and the client emitted irritating beeps as a result. I hoped that no greater irritations would come my way.

PRODUCTS, TEST SETS AND PLATFORM

The deadline for the submission of products for this review was 4 July 2005 – unwittingly causing some chaos for reasons that will be obvious to those in the US. *NetWare* itself was installed freshly from the minimum patch files provided on *Novell*'s site, for both client and server on 29 June 2005. Thus the version of *NetWare* used was *Novell Open Enterprise Server NetWare 6.5 Support Pack Revision 03, Server Version 5.70.03. NetWare Client version 4.91.0.20050216* was used on *Windows XP Professional Service Pack 2*. The client and server were connected over a 100Mbs LAN link.

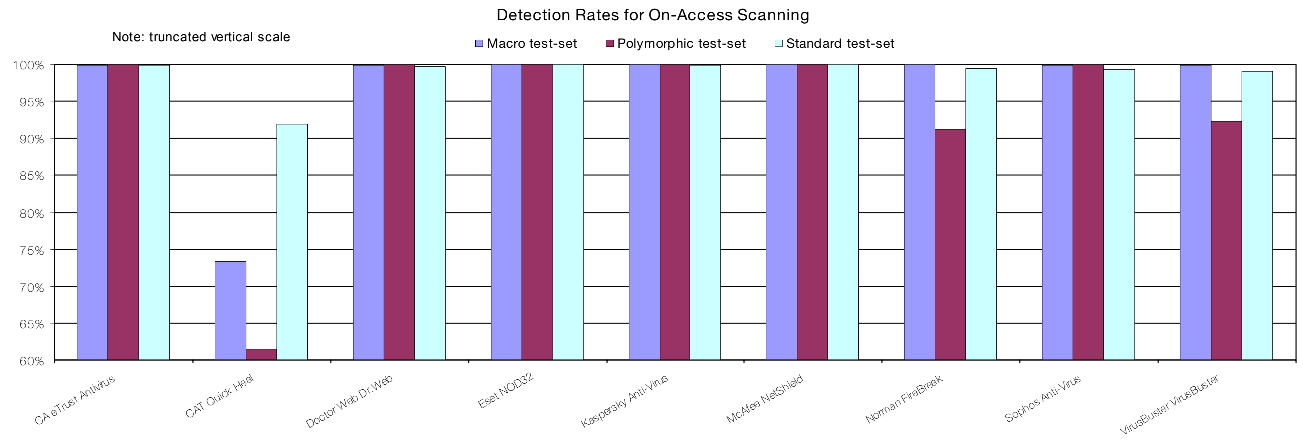
The test sets were based on the April 2005 WildList, since this was the most up-to-date version available at the time. As has been noted in recent comparative reviews, the new additions to the WildList seem to become more tedious on every occasion, though they increase numerically as if to compensate. With the new additions closing in on the 100 mark, there was only one that was not a direct variant of a sample already contained in the sets – W32/Serflog.

The majority of the new additions to the In the Wild (ItW) test set were multiple variants of W32/Sdbot and W32/Mytob. With decent handling of archives and some care in creating generic detections, these variants can, in many cases, be detected as soon as they are produced. Therefore, it seemed from the outset that simply having a *NetWare* product would almost be enough for a developer to gain a VB 100% award.

CA eTrust Antivirus 7.1

ItW File	100.00%	**Macro**	99.82%
ItW File (o/a)	100.00%	**Macro (o/a)**	99.82%
Standard	99.96%	**Polymorphic**	99.95%

eTrust is a useful example of the two facets of administration where *NetWare* products are concerned. The two main methods are to administer from a GUI (either on a client or server) or simply to interact in the server console. The latter tends to look very archaic compared with the usual interfaces for such software. In the case of *eTrust*, the on-demand scanning can be controlled fully through the



On-access tests	ItW File		Macro		Polymorphic		Standard	
	Number missed	%	Number missed	%	Number missed	%	Number missed	%
CA eTrust Antivirus	0	100.00%	12	99.82%	1	99.95%	3	99.84%
CAT Quick Heal	0	100.00%	1069	73.35%	5807	61.44%	178	91.95%
Doctor Web Dr.Web	0	100.00%	4	99.90%	0	100.00%	3	99.69%
Eset NOD32	0	100.00%	0	100.00%	0	100.00%	0	100.00%
Kaspersky Anti-Virus	0	100.00%	0	100.00%	0	100.00%	3	99.85%
McAfee NetShield	0	100.00%	0	100.00%	0	100.00%	0	100.00%
Norman FireBreak	0	100.00%	0	100.00%	180	91.24%	12	99.45%
Sophos Anti-Virus	0	100.00%	8	99.80%	0	100.00%	15	99.30%
VirusBuster VirusBuster	1	99.80%	7	99.88%	162	92.31%	17	99.01%

server console. This may also be controlled through an administration tool on a client. If full control of on-access scanning is required, however, this must be performed from the client.

Having been somewhat confused by this division of control options, the actual scanning processes were easy by contrast. Even better, *NetWare* logging is free from those strange formats which plague the *Windows* versions of *eTrust*. When logs were parsed there were no real surprises and a VB 100% award was the result.

CAT Quick Heal Antivirus 8.00

ItW File	100.00%	**Macro**	98.18%
ItW File (o/a)	100.00%	**Macro (o/a)**	73.35%
Standard	96.54%	**Polymorphic**	95.93%

Installation of *Quick Heal* is by a client-side installation routine, though the same effect may be obtained manually with little trouble. Along with this simplicity of installation, the interface is simple both in appearance (it operates through the server console) and in the limited number of options available. All the usual options are present, it is simply that they are more conveniently grouped than in many products and are not obscured by components of dubious value. Admittedly, this feeling of a lack of clutter is much helped by the fact that the on-demand and on-access components are separate NLMs. Offsetting the clarity somewhat was the log file, which changed the cases of filenames and reduced long file names to 8+3 format, somewhat hindering extraction of test results.

In fact, of all the products in this test, the results for *Quick Heal* showed the most variation between on access and on demand. Despite this, however, *Quick Heal* detected all the samples in the ItW test set, and generated no false positives, and a VB 100% award is thus due.

Doctor Web Dr.Web 4.32c (4.32.3.06300)

ItW File	100.00%	**Macro**	99.90%
ItW File (o/a)	100.00%	**Macro (o/a)**	99.90%
Standard	99.69%	**Polymorphic**	100.00%

Doctor Web's *NetWare* product remains essentially the same in look and feel as when I inspected it several years ago. Setting it up is performed simply by copying the files to the server and loading the NLM. This either results in a working interface or exits with the reason for

On-demand tests	ItW File		Macro		Polymorphic		Standard	
	Number missed	%	Number missed	%	Number missed	%	Number missed	%
CA eTrust Antivirus	0	100.00%	12	99.82%	1	99.95%	1	99.96%
CAT Quick Heal	0	100.00%	75	98.18%	418	95.93%	101	96.54%
Doctor Web Dr.Web	0	100.00%	4	99.90%	0	100.00%	3	99.69%
Eset NOD32	0	100.00%	0	100.00%	0	100.00%	0	100.00%
Kaspersky Anti-Virus	0	100.00%	0	100.00%	0	100.00%	0	100.00%
McAfee NetShield	0	100.00%	0	100.00%	0	100.00%	0	100.00%
Norman FireBreak	0	100.00%	0	100.00%	180	91.24%	12	99.45%
Sophos Anti-Virus	0	100.00%	0	100.00%	0	100.00%	12	99.45%
VirusBuster VirusBuster	1	99.80%	7	99.88%	151	92.62%	14	99.31%

failure dumped to a log. The lack of an on-screen message to inform me that the licence key was not found, caused me a little perplexity until I found this log. However, once installed all went smoothly.

Scanning results were much the same as have been noted in recent *Windows* testing. *Dr.Web* seems to alternate between full detection and missing a small number of samples – the latter presumably being due to the tweaking of older definitions for efficiency. No misses occurred in the ItW set, however, and with no false positives *Dr.Web* receives a VB 100%.

Eset NOD32 1.11.61

ItW File	100.00%	**Macro**	100.00%
ItW File (o/a)	100.00%	**Macro (o/a)**	100.00%
Standard	100.00%	**Polymorphic**	100.00%

Likewise unchanged since the last few tests, the on-demand and on-access scanners of *NOD32* are each comprised of an NLM which is loaded from the server console. The word 'loaded' is perhaps a little misleading in the case of the on-demand scanner which, alone in these tests, operates as a command-line scanner rather than having any more advanced interface.

This rather aged interface might cause second thoughts for some users. The full detection rates and good scanning speed, however, can cause no such issues and result in a further VB 100% award for *Eset*.

Kaspersky Anti-Virus 5.6.1

ItW File	100.00%	**Macro**	100.00%
ItW File (o/a)	100.00%	**Macro (o/a)**	100.00%
Standard	100.00%	**Polymorphic**	100.00%

The *Kaspersky* product is rather more evolutionarily advanced than some others, the default installation from the client being one sign of this. It installs as a snap-in to ConsoleOne, *Novell*'s *NetWare* GUI. After installation there are two server console interfaces, one each for the on-demand and on-access scans. These are, however, informational rather than interactive, and scanning during testing was controlled via the ConsoleOne interface. Logging proved somewhat confusing for a while, until it became clear that the use of ampersands in file names was causing the log entries to become garbled.

With the log files unravelled there was a small difference in results between the on-access and on-demand tests, with the latter showing full detection. However, the files missed on access were due to the understandable removal of archive handling for files in this mode – a common efficiency measure. None of the files missed were in the ItW test set and thus *Kaspersky* receives another VB 100% in this month's bumper crop.

McAfee NetShield 4.6.3 4.4.00 4.0.4529

ItW File	100.00%	**Macro**	100.00%
ItW File (o/a)	100.00%	**Macro (o/a)**	100.00%
Standard	100.00%	**Polymorphic**	100.00%

The installation of *NetShield* was delayed a little by the requirement for a Java runtime to be available on the machine from which the install will take place. Once this hurdle had been overcome, the process of installation from a client was simple enough. Updates and upgrades were applied to the software by the expedient of unloading the NLMs and overwriting old files with new – which seems to be a common method in *NetWare*.

The main NLM for *NetShield* operates as a server console-viewable interface, though it can only be inspected in this state. In order to adjust the configuration, the client side application must be used. This offers exactly the same interface as *NetShield* on other platforms. The developers seem to have opted for minimising network traffic during scanning, since despite having a scan status visible in the GUI, this status was not updated between the start and end point of any scan.

With no samples missed in any of the test sets, and no false positives generated in the clean set, *McAfee* is due a VB 100% award without further ado.

Norman FireBreak 4.74 2311

ItW File	100.00%	**Macro**	100.00%
ItW File (o/a)	100.00%	**Macro (o/a)**	100.00%
Standard	99.45%	**Polymorphic**	91.24%

The installation procedure for *FireBreak* is performed from the client, requiring a drive to be mapped to the root of SYS: on the server. A ConsoleOne snap-in and Internet update module are installed as part of this process, though the server console interface was used for testing.

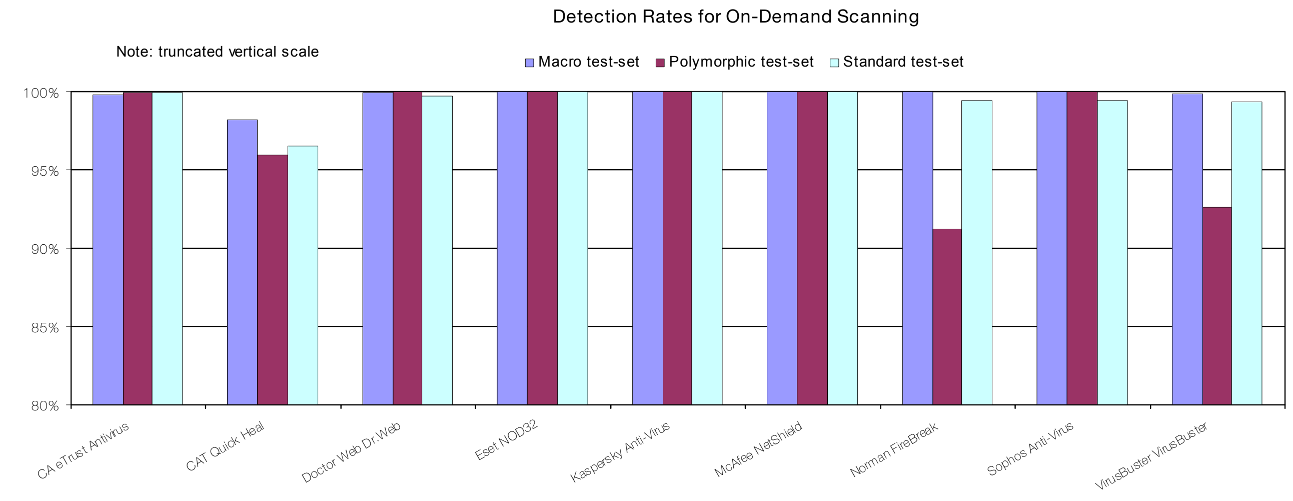
On the occasion of the last review, there were a number of problems for *Norman*'s product, associated with scanning. Thankfully these were notable only by their absence this time.

The detection rate was very much at the level usually achieved by *Norman*. Weaknesses still exist in the handling of relatively modern polymorphic viruses, though none of these were present in the ItW test set. A VB 100% award is the net result.

Sophos Anti-Virus 3.95.0

ItW File	100.00%	**Macro**	100.00%
ItW File (o/a)	100.00%	**Macro (o/a)**	99.80%
Standard	99.45%	**Polymorphic**	100.00%

Another product adhering firmly to the server console style of interface, *Sophos Anti-Virus* is also very much unchanged by the passage of time. Installation is by the loading of a single NLM, which creates the appropriate



Hard Disk Scan Rate	Executables			OLE Files			Zipped Executables		Zipped OLE Files	
	Time (s)	Throughput (kB/s)	FPs [susp]	Time(s)	Throughput (kB/s)	FPs [susp]	Time (s)	Throughput (kB/s)	Time(s)	Throughput (kB/s)
CA eTrust Antivirus	267.0	2048.4		15	5288.9		130.0	1226.3	15.0	4973.8
CAT Quick Heal	140.0	3906.7		13	6102.6		76.0	2097.6	19.0	3926.7
Doctor Web Dr.Web	180.0	3038.5		16	4958.4		31.0	5142.5	5.0	14921.5
Eset NOD32	75.0	7292.4		9	8814.9		20.0	7970.8	4.0	18651.9
Kaspersky Anti-Virus	285.0	1919.1		29	2735.6		116.0	1374.3	27.0	2763.2
McAfee NetShield	420.0	1302.2		29	2735.6		265.0	601.6	31.0	2406.7
Norman FireBreak	248.0	2205.4		14	5666.7		22.0	7246.2	5.0	14921.5
Sophos Anti-Virus	152.0	3598.2		17	4666.7		18.0	8856.5	7.0	10658.2
VirusBuster VirusBuster	621.0	880.7	[1]	25	3173.4		207.0	770.1	20.0	3730.4

directories and populates them. This is a convenient set up procedure, which avoids the irritation of setting search paths and directory structures. Having added supplementary virus identities the product is ready for operation.

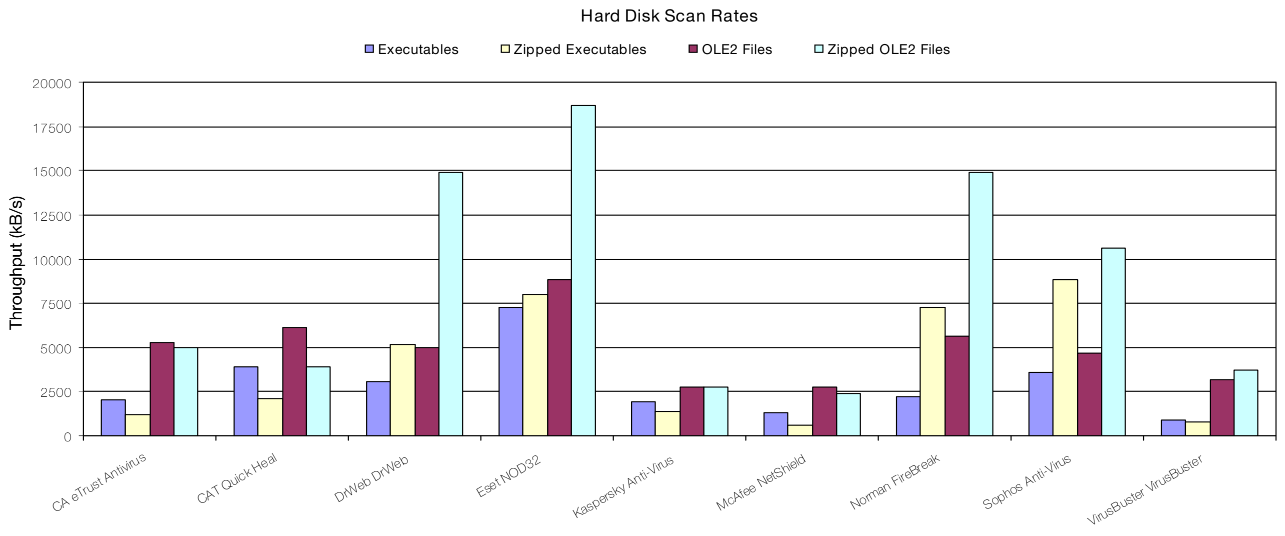
Age-old niggles still exist during operation, however. The requirement to prepend '>' to paths in order to force recursive scanning is among the more idiosyncratic parts of the interface. The log file is now out of step even with other *Sophos* products, still reducing long file names to the less than useful '?????~?.???' format. It should be noted that it is impossible to scan anything other than a full volume using the extension lists supplied, thus the scanning here was performed on all files in a supplied path. Despite these peculiarities the scanning performed without any hitches and resulted in a full detection of ItW files. A VB 100% is thus secured by *Sophos*.

VirusBuster VirusBuster 2005 v2.02.003

ItW File	99.80%	**Macro**	99.88%
ItW File (o/a)	99.80%	**Macro (o/a)**	99.88%
Standard	99.31%	**Polymorphic**	92.62%

VirusBuster installs by copying its files to the server, setting the location as a search path and loading the main NLM.

The main issue with *VirusBuster* concerned its speed of scanning infected files. This was noticeably slow in the ItW



test set, though this is common enough with the unpacking required for some of the bot samples in the collection.

Rather more frustrating were some polymorphic samples. In particular, Satanbug.5000.A took over a minute per sample to be scanned in many cases. With 500 samples of this virus alone in the test sets, scanning was a time-consuming and tedious process indeed. On the plus side, the *VirusBuster* logs now make a distinction between worms and viruses, though with the eternal debate over the fine distinctions of the nomenclature, this may only serve to inflame passions.

VirusBuster demonstrated the only false positive in the tests, although this was simply a sample which was declared suspicious rather than a full-blown declaration of viral content. Unfortunately, however, *VirusBuster* missed the W32/Lovelorn.A sample in .HTM form both on access and on demand. As this sample is in the wild, *VirusBuster* misses out on a VB 100% on this occasion.

CONCLUSION

Looking back over the last few *NetWare* reviews (see for example *VB*, August 2004, p. 14 and *VB*, August 2003, p.17) I find myself repeating my comments, especially concerning the two broad groups into which the developers have fallen. On the one hand some developers continue to add to their products administrative functionality and integration within a managed anti-virus environment. On the other hand there are those whose only developmental effort seems to have been in making the product detect more viruses, with all other features remaining in stasis.

NetWare itself seems in a healthier state than it has been in the recent past, with *Novell*'s strategic partnerships being chosen to bring the company out of the dark corner into which it was pushed by other server offerings. Whether this will be enough to encourage further anti-virus developer effort remains to be seen.

Technical details:

Test environment: Identical 1.6 GHz *Intel Pentium* machines with 512 MB RAM, 20 GB dual hard disks, DVD/CD-ROM and 3.5-inch floppy drive. Server running *Novell Open Enterprise Server NetWare 6.5 Support Pack Revision 03*, Server version 5.70.03. Client running *Novell NetWare Client* version 4.91.0.20050216 installed on *Windows XP Professional Service Pack 2.*

Virus test sets: Complete listings of the test sets used are at http://www.virusbtn.com/Comparatives/NetWare/2005/test_sets.html.

A complete description of the results calculation protocol is at http://www.virusbtn.com/Comparatives/Win95/199801/protocol.html.

END NOTES & NEWS

The IDC Security Conference takes place on 14 September 2005 in London, UK. Delegates will hear how other organisations have ensured the security of their business through the use of technology and security strategies. See http://www.idc.com/uk/security05/.

The Gartner IT Security Summit takes place 14–15 September 2005 in London, UK. The summit will look at how current technology provides guidance on which old and new product categories are most useful in controlling information security risk. For more information see http://www.gartnerinfo.com/.

T2'05, the second annual T2 conference, will be held 15–16 September 2005 in Helsinki, Finland. The conference focuses on newly emerging information security research. All presentations are technically oriented, practical and include demonstrations. See http://www.t2.fi/english/.

COSAC 2005, the 12th International Computer Security Symposium, takes place 18–22 September 2005 near Dublin, Ireland. A choice of more than 40 sessions and six full-day master classes and forums is available. The full programme and details of how to register are available at http://www.cosac.net/.

The Network Security Conference takes place 19–21 September 2005 in Las Vegas, NV, USA. The conference is designed to meet the education and training needs of the seasoned IS professional as well as the newcomer. For details see http://www.isaca.org/.

The 5th Annual FinSec Conference takes place 20–23 September 2005 in London, UK. This year's conference will focus on the unique set of challenges afflicting information security professionals in the financial community. See http://www.mistieurope.com/.

The 4th annual SecurIT Summit will be held 28–30 September 2005 in Montreux, Switzerland. SecurIT 2005 will integrate a busy conference programme, one-to-one business meetings and informal networking with leisure activities. For more information see http://www.securit-summit.com/.

e-Secure Malaysia 2005 takes place 28 September to 1 October 2005 in Kuala Lumpur, Malaysia. The exhibition and conference will cover issues such as computer emergency response, spam and viruses, hacking, cyber laws and terrorism, security management, access control and network security. See http://www.protemp.com.my/.

The SophosLabs Malware Analysis Workshop will be held 4 October 2005. The course is aimed at IT security professionals who are responsible for implementing and maintaining IT security solutions, or who are involved in computer security research. For details see http://www.sophos.com/.

The 15th Virus Bulletin International Conference, VB2005, will take place 5–7 October 2005 in Dublin, Ireland. The programme for the three-day conference can be found on the *VB* website. For more information or to register online see http://www.virusbtn.com/.

Black Hat Japan (Briefings only) will be held 17–18 October 2005. See http://www.blackhat.com/.

RSA Europe 2005 will be held 17–19 October 2005 in Vienna, Austria. For more details see http://www.rsaconference.com/.

WORM 2005 (the 3rd Workshop on Rapid Malcode) will take place 11 November 2005 in Fairfax, VA, USA. The workshop will provide a forum to bring together ideas, understanding and experiences bearing on the worm problem from a wide range of communities, including academia, industry and the government. For more details see http://www1.cs.columbia.edu/~angelos/worm05/.

The eighth Association of Anti-Virus Asia Researchers International Conference (AVAR 2005), takes place in Tianjin, China on 17 and 18 November 2005. The theme of this year's conference will be 'Wired to Wireless, Hacker to Cybercriminal'. For more details email avar2005@antivirus-china.org.cn or see http://aavar.org/.

Infosecurity USA will be held 6–8 December 2005 in New York, NY, USA. The conference will take place 6–8 December, with the accompanying exhibition running from 7–8 December. The full conference programme will be announced this month. For details see http://www.infosecurityevent.com/.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues) including first-class/airmail delivery: £195 (US$358)

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire OX14 3YP, England

Tel: +44 (0)1235 555139 Fax: +44 (0)1235 531889
Email: editorial@virusbtn.com Web: http://www.virusbtn.com/

vb Spam supplement

CONTENTS

S1	**NEWS & EVENTS**
S1	**FEATURE** Exploiting vulnerabilities in the human interface for fun and profit

NEWS & EVENTS

SPAMMER REFORMED?

Scott Richter, aka the 'Spam King', is no longer classed as a spammer according to *Spamhaus*'s authoritative Register of Known Spam Operations (ROKSO). The man who brought to our inboxes such 'unbeatable' offers as the set of Iraq's Most Wanted playing cards for only $5.99 has apparently decided to go straight, ditching the spam in favour of legitimate opt-in bulk mailing.

Richter gained notoriety as a spammer thanks to a catalogue of high-profile campaigns and legal cases. In May 2004 Richter was granted a restraining order against spam-reporting outfit *SpamCop*. The following month, shortly after he began to market a range of clothing under the 'Spam King' label, Richter received a cease-and-desist communication from *Hormel Foods*, warning him against using the Spam marque. July 2004 saw Richter agree to pay $40,000 in settlement of a lawsuit brought against him by the State of New York, and in March this year his email marketing company *OptInRealBig.com* was brought to the brink of bankruptcy by a *Microsoft* lawsuit. On filing for Chapter 11 bankruptcy protection, Richter claimed company assets of less than $10 million and debts of more than $50 million.

Spamhaus director Steve Linford reported that a significant drop in overall spam levels has been noted over the six months since Richter gave up spamming.

EVENTS

TREC 2005, the Text Retrieval Conference, will be held 15–18 November 2005 at NIST in Gaithersburg, MD, USA. For more details see http://trec.nist.gov/.

FEATURE

EXPLOITING VULNERABILITIES IN THE HUMAN INTERFACE FOR FUN AND PROFIT

Heather Goudey
Computer Associates, Australia

In order to be successful, spammers, scammers and malware writers must construct messages that are sufficiently compelling to motivate their targets to act in the desired manner (e.g. to execute an attachment, or click on a URL and enter their banking details on a spoofed website).

The Internet and supporting technologies provide a low-cost, far-reaching and effective means of deceiving people on an unprecedented scale. Regardless of the development and availability of sophisticated security applications and network hardening technologies, humans are still touted to be the platform-independent vulnerability to every system, the so called 'weakest link' in the security chain.

It appears that people exhibit certain behaviours that render them vulnerable to exploitation by others who are so motivated. Users are targeted by many persuasive messages and appeals every day, and have limited cognitive resources to expend on judging which messages are legitimate. This necessitates the use of behavioural heuristics that are not necessarily as useful for determining legitimacy in emails as they have proved in the past for more traditional communication channels (such as, face to face, or other so-called 'rich' media).

This article examines email as a communication medium and channel for malware and malicious content, and provides a brief meta-analysis of the current research regarding social engineering and deception theory with a view to ascertaining why users act on the receipt of deceptive messages.

EMAIL

Email is a widely used, high-speed, cheap, asynchronous communication medium that has virtually unlimited range and gives the perpetrators of attacks via this channel easy access to very large numbers of targets [1, 2].

Email also lends itself to these types of attack, thanks to the relative ease with which addresses can be masked or created. Often an address is spoofed to appear to come from a trusted source. Senders take advantage of existing trust relationships with third parties, where a receiver is significantly more likely to execute even an unexpected attachment from a trusted and familiar source.

Thoughtlessness is another issue. With many messages to scan through every day, users are likely to make a number of errors during the sorting process. Attachments are used so widely and legitimately that users give little, if any thought to executing them. While telling your users never to open any attachments unless they are expected may seem like great advice, in reality this just results in large numbers of 'false positives' [3].

Email has traditionally been thought of as a 'lean' communication medium. One popular theory that has developed to explain differences in media is that of media richness, first expounded by Daft in 1984 [4]. This theory suggests that a communication medium can be described in terms of its 'richness'; that is, its capacity to carry information. According to media richness theory, media that has higher degrees of speed of feedback, cue multiplicity, language variety and personal focus is considered to be richer [5].

There is a belief that email falls at the lower end of the richness scale and that this medium is incapable of transmitting many of the cues that are available via richer communication channels. However, anecdotal evidence (estimates regarding the cost of phishing or the distribution of worms that spread via email, for example) suggests that even without richer cues, email is still a viable channel for deceptive communications.

SOCIAL ENGINEERING

Definitions of social engineering in the current literature are inconsistent at best. Harley [6] defines it as the 'psychological manipulation, skilled or otherwise, of an individual or set of individuals to produce a desired effect on their behaviour', while others have referred to it rather simply as 'a hacker's clever manipulation of the human tendency to trust' [7].

While these definitions go some of the way to defining social engineering, a more formal definition may prove more useful [8]: 'Social engineering techniques are used to target the human interface (as opposed to hardware and/or software) in order to compromise data and/or systems [for the attacker's benefit]. These techniques are comprised of a two-step approach that initially deceives targets to perceive a false reality, and then takes advantage of particular social psychological traits to convince targets to act on the misinformation contained in the message.'

Hence, social engineering is really a particular subset of a much broader and more formal area of research: deceptions and social psychology [9]. Luckily, while the study of deception via Computer Mediated Communication (CMC) may be fairly recent, a review of this comprehensive and formalized body of research may prove useful in this context.

DECEPTION

People lie often and regularly, and deception plays a significant role in day-to-day information exchange. Research expounds that most deceptive communication takes place via rich media channels, such as face to face or voice, and that while people are able to detect deception only a little better than chance, those cues that have been discovered to enable people to detect the attempt to deceive have been non-verbal – such as pupil dilation, blinking, pitch of voice, etc. [1, 10, 11].

Logic leads one to the conclusion that deception may be more difficult using less rich channels (e.g. email), but spamming techniques make this logic a little fuzzy – a very small response (or success) rate from spamming a very large audience is still a viable and profitable success for the perpetrator of such attacks. Malware and malicious content distributors don't need to fool every target.

Deception may be defined as 'a message knowingly transmitted by a sender to foster a false belief or conclusion by the receiver' [9]. The sender/deceiver manipulates the information in a message in order to persuade the receiver to alter their behaviour to the benefit of the sender/deceiver [12].

This approach is strengthened by a behavioural characteristic known as truth bias, where users inherently tend to accept what they are told as truthful. Unless trained or otherwise conditioned, users are unlikely to approach messages with suspicion, or the expectation that they are deceptive [13].

In his paper, 'Toward a Theory of Deception' [14], Bowyer outlines a formalised process by which deception is performed. Whilst most of Bowyer's analogies are militaristic in nature, they are still highly appropriate to our context of malware and other malicious content distributed via email.

While Bowyer goes into a lot more detail, a simplified and applied version of his process includes:

1. Deception planning: this includes the deceiver's objective goal, possible costs and benefits to the

deceiver should the ruse be successful, and channel (or media) selection. For example, with the objective of stealing users' online banking details, a deceiver plans to spam out a phishing email that attempts to deceive users and encourage them to enter this sensitive information to a spoofed banking site.

2. Ruse construction: the characteristics that must be combined to create a successful ruse. In our example, this would include constructing a convincing email that appears to come from the targeted bank, locating a machine (possibly via compromise) to host the spoofed site, having access to a large email distribution list, having access to machines to perform the spam run, disguising URLs, etc.
3. The ruse is channelled into the decision arena – in our example, the channel is email, and the spam run begins. At this point, the target receives the deceptive message and either believes the ruse and acts in the manner suggested by the deceiver, believes the ruse and does nothing, or does not believe the ruse. At this point, the ruse and its success is out of the hands of the deceiver and there are a lot of other factors that may come into play and affect the final outcome. The ruse interacts with objective reality.

   In our example, the target receives the deceptive phishing email and uses their experience with the channel, their knowledge of the organizational context of the message and what they know about the spoofed sender in order to decide how to act on receipt of the message.

   This interaction with reality can be described by 'channel expansion', a theory that expands on media richness theory to take into account additional and contextual interactions with a communication. The more experience that users have with the channel, the richer they find the media they are using and the more they may find that they are able to detect the deception [10]. A canny and experienced email recipient would be more likely to have heard of phishing scams and more likely to spot the ruse as the deception it really is.
4. Feedback: the receiver's response to the illusion as received by the deceiver – in this case, whether the targeted users did indeed enter their details to the spoofed site for later collection.

When it comes to creating a convincing ruse, deception can be broken into two types: hiding the real, and showing the false.

These groups can be further broken down into subgroups, where hiding encompasses methods that use masking, repackaging, or dazzling, while showing encompasses mimicking, inventing and decoying. These methods may be used to create convincing ruses that copy reality, create new realities or subvert the existing reality [12, 14].

Regardless, the actions of human beings depend upon the process of interpreting their perceived reality. Decisions are made within the context of this reality. Persuasive appeals may make use of this distorted context in combination with exploiting human behaviours in order to convince users to act.

This is further compounded by a concept known as 'bounded rationality', a theory that humans have limited cognitive resources with which to make decisions, and as such seek an acceptable trade-off between cost and satisfaction [3]. Not every decision is made in a considered manner, and an answer that is 'good enough' is often the best answer that can be expected from users. These behavioural heuristics, or shorthand, while generally very useful, can also be exploited by the use of particular persuasive techniques.

While little examined, research regarding social engineering as a subset of deception theories may prove very useful when applied to the study of user responses to the distribution of malware and malicious content. The meta-analysis provided here is little more than a taste of the more basic concepts involved in this and related disciplines, and the 'further reading' section listed below is highly recommended.

With so much email proving to be deceptive, a broader, cross-discipline approach may be warranted. If we understood more about why even experienced users respond to artifice, then we could test new approaches for mitigation. A new channel for malicious content does not necessarily mean that we need to throw out existing research and start again.

Although little has been stated here regarding the solutions proposed by research into deception theory, it is important to note that this research is more than purely academic. Several proponents of this discipline have published research that tests approaches to training users in order to detect deception, the intent to deceive or the intent to persuade illegitimately and on automated statistical analysis of deceptive text.

Deception is part of the natural order of human behaviour. You would think, considering how long we have been subjected to deception, that we would have come equipped with better resources to detect it and react appropriately, regardless of the novelty of the channel used to distribute deceptive messages.

While humans might not necessarily be the 'weakest link' in the security chain, their limitations with regard to limited cognitive resources and truth bias are worthy of consideration. Technologies that remove humans from contact with deceptive messages (such as anti-virus and anti-spam) are particularly useful given these limitations. Given the number of deceptive messages that bombard users' inboxes daily, a greater harnessing and elaboration of theories and experiments regarding deception in this context are certainly warranted.

REFERENCES

[1] George, J.F., and Carlson, J.R, 'Group support systems and deceptive communication', *Proceedings of the 32nd Hawaii International Conference on System Sciences*, 1999.

[2] Grazioli, S., and Wang, A., 'Looking without seeing: understanding naïve consumers' success and failure to detect Internet deception', *Proceedings of the International Conference on Information Systems*, 2001.

[3] Besnard, D., and Arief, B., 'Computer security impaired by legitimate users', *Computers & Security*, Volume 23, Issue 3, pp.253–264, 2004.

[4] Daft, R.L., and Lengel, R.H., 'Information richness: a new approach to managerial behavior and organizational design', in: Cummings, L.L. & Staw, B.M. (Eds.), *Research in Organizational Behavior 6*, pp.191–233, Homewood, IL: JAI Press, 1984.

[5] Daft, R.L., and Lengel, R.H., 'Organizational information requirements, media richness and structural design', *Management Science* 32(5), pp.554–571, 1986.

[6] Harley, D., 'Re-floating the Titanic: dealing with social engineering attacks', *EICAR Conference* 1998.

[7] Granger, S., 'Social engineering fundamentals, part 1: hacker tactics', *Security Focus* 18/12/2001. Retrieved 6 January 2005, from http://www.securityfocus.com/infocus/1527/.

[8] Jordan, M., and Goudey, H., 'The signs, signifiers and semiotics of the successful semantic attack, *EICAR Conference* 2005.

[9] George, J.F., Marett, K., Burgoon, J.K., Crews, J., Cao, J., Lin, M., and Biros, D.P., 'Training to detect deception: an experimental investigation', *Proceedings of the 37th Hawaii International Conference on System Sciences*, 2004.

[10] Zhou, L., Twitchell, D.P., Qin, T., Burgoon, J., and Nunamaker, J.F.Jnr, 'An exploratory study into deception detection in text-based computer mediated communication', *Proceedings of the 36th Hawaii International Conference on System Sciences*, 2003.

[11] Keila, P.S., and Skillicorn, D.B., 'Detecting unusual and deceptive communication in email', external technical report, School of Computing, Queen's University, Kingston, Ontario, Canada, 2005.

[12] Hutchinson, W., and Warren, M.J., 'Nothing is real: deception in the information age', *Journal of the Australian Institute of Professional Intelligence Officers*, 9(1) 27–40, 2001.

[13] Boyle, R.J., and Ruppel, C.P., 'The impact of media richness, suspicion and perceived truth bias on deception detection', *Proceedings of the 38th Hawaii International Conference on System Sciences*, 2005.

[14] Bowyer Bell, J., 'Toward a theory of deception', *International Journal of Intelligence and Counterintelligence*, 16, pp.244–279, 2003.

FURTHER READING

[i] George, J.F., and Carlson, J.R, 'Media selection for deceptive communication', *Proceedings of the 38th Hawaii International Conference on System Sciences*, 2005.

[ii] Kiesler, S., Siegel, J., and McGuire, T., 'Social psychological aspects of computer mediated communications', *American Psychologist*, 39, pp.1123–1134, 1984.

[iii] Mitchell, R.W., 'The psychology of human deception – truth-telling, lying and self-deception', *Social Research*, Fall 1996. Retrieved 3 January 2005, from http://www.findarticles.com/p/articles/mi_m2267/is_n3_v63/ai_18888994/.

[iv] Rusch, J., 'The "Social Engineering" of Internet Fraud', 1999. Retrieved 4 January 2005, from http://www.isoc.org/isoc/conferences/inet/99/proceedings/3g/3g_2.htm..

[v] Sagarin, B.J., Serna, S.B., Cialdini, R.B., and Rice, W.E., 'Dispelling the illusion of invulnerability: the motivations and mechanisms of resistance to persuasion', *Journal of Personality and Social Psychology*, Vol 83, pp.526–541, 2002.

[vi] Spears, R., Postmes, T., Wolbert, A., Lea, M., and Rogers, P., 'Social psychological influence of ICTs on society and their policy implications', in R. Van der Ploeg & C. Veenemans (Eds.) *Amsterdam Infodrome*: 3–84, 2001.