FEBRUARY 2013

virus BULLETIN

Fighting malware and spam

CONTENTS

2 **COMMENT**
Targeted attacks: what's in store?

3 **NEWS**
VB2014: location, location, location
Results of cybersecurity exercise published

3 **MALWARE PREVALENCE TABLE**

MALWARE ANALYSES
4 A(C)ES high
7 Please help!

9 **FEATURE**
Techniques for evading automated analysis

COMMENTARIES
13 Anti-virus: last rites, or rites of passage?
15 Breaking down barriers for cybersecurity: where's the first-mover advantage?

19 **END NOTES & NEWS**

ISSN 1749-7027

IN THIS ISSUE

NEW INTEREST

In 2009, Intel introduced a new set of CPUs that included hardware support for the Advanced Encryption Standard (AES) in the instruction set. These were not in wide circulation until relatively recently, and consequently have not attracted much interest from virus writers – until now. Peter Ferrie describes the W32/Brotinn virus.

page 4

VIRTUAL EVASION

Abhishek Singh looks at some of the techniques that are commonly used by malware to bypass analysis in a virtualized environment.

page 9

DEAD AGAIN

While some are claiming that AV is so far past its best-before date that it should only be used when given away free, David Harley asks on what basis this judgement has been made, and whether the reality is that anti-virus is simply no longer the same product as it was decades ago.

page 13

VIRUS BULLETIN COMMENT

‘The general level of insight into network infiltration around the globe is becoming more informed.’

Kurt Baumgartner
Kaspersky Lab

TARGETED ATTACKS: WHAT’S IN STORE?

Targeted attacks, determined adversaries, or the APT. Whatever label you use, this is not a new topic, but clearly the general level of insight into the reality of network infiltration around the globe is becoming more informed. There has been at least some level of public discussion about each of the following attacks from the past year: Red October, Madi, miniFlame/SPE, Gauss, Flame, Enfal, Voho, Elderwood, and various Comment Crew attacks. More details are being presented to the public, and this is progress.

The recently reported Red October attack was unprecedented in the breadth and scope of its sustained level of occupation within diplomatic targets, heavily funded research organizations, military interests and more. This was an advanced cyber-espionage campaign that collected geo-political intelligence. The Red October crew poured out a customized toolset to penetrate deeply, blend into their targets and reach beyond. We hadn’t previously seen resurrection modules used by plug-in components entrenched in embassy networks around the world, which were prepared to be discovered and then re-entrench from the victim systems themselves. We hadn’t seen modules customized like these to suck data from individual mobile manufacturers’ devices and retrieve contacts and data. To date, we have not had fully comprehensive information presented in an organized fashion on large-scale, targeted threats. It required months of effort to collect and research the full Red October toolset, and both interesting components and changes in the components over time and per victim continue to be uncovered. For the first time, a full list of indicators based on the OpenIOC format has been released to coincide with the large Red October public release for CERTs, network admins and legitimately interested parties. Perhaps this exhaustive report is helping to move real discussion and action forward in concrete terms that have not been available during previous incidents that were more likely pushed to generate marketing buzz than for any other purpose.

What else has changed over the past year in relation to targeted attacks? In the US, SEC guidance passed approximately a year ago was supposed to push forward public discussion and investor awareness. Unfortunately, timely, informative breach reports have not materialized. A couple of exceptions come to mind, including *Adobe*’s, but for the most part, organizations with breached networks (and their contractors with breached networks) seem to continue to hide or ignore the problem. On the technical side, *Flash* and *Reader* seem to be on the decline as exploitation targets at victim organizations, having been replaced with *Office* and Java targets. Defensive technologies and programs have improved, and public discussion around these attacks cannot be ignored at this point.

So what is in store for us this year? Offensive campaigns show no sign of letting up. Attackers will improve their toolsets, and mobile devices will come to light as an initial vector for targeted attack payloads. The demand to access data in the cloud from mobile devices as well as standard workstation/laptop devices will be exploited by the APT. Portions of various cloud implementations will be breached. Overwhelmed and underprepared CERTs across the globe will improve their capabilities, but prolonged absences in some countries (due to national holidays) will continue. Problems within critical infrastructure security will be more widely attacked – and discussed. For better or worse, some victim organizations will attempt to ‘hack back’, and full attribution and active defence will be better used and understood. Potential victims and targeted organizations will be incentivized to share data. Various categories of non-corporate victims will talk more freely about their incidents, especially human rights organizations. The concept of ‘sophistication’ will be replaced within media reports with the concept of ‘efficacy’, and quibbling over the term ‘advanced’ will finally exhaust itself. Whichever way you cut it, there will be an increased level of targeted activity this year.

Editor: Helen Martin
Technical Editor: Dr Morton Swimmer
Test Team Director: John Hawes
Anti-Spam Test Director: Martijn Grooten
Security Test Engineer: Simon Bates
Sales Executive: Allison Sketchley
Perl Developer: Tom Gracey
Consulting Editors:
Nick FitzGerald, *AVG, NZ*
Ian Whalley, *Google, USA*
Dr Richard Ford, *Florida Institute of Technology, USA*

NEWS

VB2014: LOCATION, LOCATION, LOCATION

Keeping shtum about the dates and details for VB2014 – the 24th Virus Bulletin International Conference – has not been an easy task, so it is with delight and relief that, with contracts in place and legal teams satisfied, we can now announce that VB2014 will take place 24–26 September 2014 at The Westin Seattle, WA, USA. More details will be announced in due course at http://www.virusbtn.com/conference/vb2014/.

As usual, a range of sponsorship opportunities will be available for VB2014 at Platinum, Gold and Silver sponsorship levels. If you are interested in becoming a sponsor of VB2014 or exhibiting at the event please contact us by emailing conference@virusbtn.com.

RESULTS OF CYBERSECURITY EXERCISE PUBLISHED

ENISA (the European Network and Information Security Agency) has published a report on its pan-Europe cybersecurity exercise 'Cyber Europe 2012', which it ran in October. This was the largest exercise of its kind, involving almost 600 individual players from 29 EU and EFTA member states.

The report's conclusion was that, for fast and effective response to cyber incidents, knowledge of procedures and information flows is crucial.

Among the report's key findings was the fact that cooperation and information exchange between public and private players is necessary, and that public-private cooperation structures differ between countries – with parallel, sometimes overlapping, public and private procedures on the national level presenting a challenge to national level cooperation.

The report recommended more pan-European and national cyber exercises to improve cross-border cooperation, as well as increased private sector involvement at the national level for future exercises. It recommended that countries work on improving the effectiveness, scalability and knowledge of existing mechanisms, procedures and information flows for both national and international cooperation.

The full report is available (in the 23 official EU languages) at https://www.enisa.europa.eu/activities/Resilience-and-CIIP/cyber-crisis-cooperation/cyber-europe/cyber-europe-2012/cyber-europe-2012-key-findings-report-1.

Prevalence Table – December 2012[1]

Malware	Type	%
Autorun	Worm	8.55%
Java-Exploit	Exploit	8.49%
Adware-misc	Adware	7.01%
OneScan	Rogue	6.04%
Heuristic/generic	Trojan	5.27%
Heuristic/generic	Virus/worm	4.71%
Conficker/Downadup	Worm	4.48%
Iframe-Exploit	Exploit	3.80%
Crypt/Kryptik	Trojan	3.72%
Potentially Unwanted-misc	PU	3.36%
Agent	Trojan	2.74%
Sality	Virus	2.61%
Encrypted/Obfuscated	Misc	2.59%
Sirefef	Trojan	2.11%
Downloader-misc	Trojan	2.08%
Injector	Trojan	1.87%
Dorkbot	Worm	1.85%
Zwangi/Zwunzi	Adware	1.61%
LNK-Exploit	Exploit	1.53%
Virut	Virus	1.34%
Crack/Keygen	PU	1.33%
Exploit-misc	Exploit	1.22%
Heuristic/generic	Misc	1.13%
BHO/Toolbar-misc	Adware	1.03%
Zbot	Trojan	0.99%
Qhost	Trojan	0.99%
Tanatos	Worm	0.93%
Blacole	Exploit	0.90%
Ramnit	Trojan	0.87%
Brontok/Rontokbro	Worm	0.80%
Jeefo	Worm	0.79%
JS-Redir/Alescurf	Trojan	0.77%
Others[2]		12.50%
Total		**100.00%**

[1] Figures compiled from desktop-level detections.

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

MALWARE ANALYSIS 1

A(C)ES HIGH

Peter Ferrie
Microsoft, USA

Intel introduced a new set of CPUs in 2009 that included hardware support for the Advanced Encryption Standard (AES) in the instruction set. Such CPUs were not in wide circulation until relatively recently, and as a result, they did not attract much interest from virus writers. All that has changed, however, with the release of the W32/Brotinn virus.

REGISTER NOW

The first generation of the virus begins by registering a Structured Exception Handler to intercept any errors that occur, but the entry point of the host is used as the handler procedure. This means that if an exception occurs, the Structured Exception Handler structure remains registered, which can lead to unexpected behaviour if the host expects the initial Structured Exception Handler address to be in the kernel. The virus retrieves the base address of kernel32.dll. It does this by walking the InMemoryOrderModuleList from the PEB_LDR_DATA structure in the Process Environment Block.

The virus resolves the addresses of the API functions that it requires, which is just a bit more than the bare minimum that it needs for infection: find first/next, set attributes, open, map, unmap, close and GetTickCount (which is used for seeding the random number generator). There is an interesting advancement here: the APIs in this virus are Unicode-only, instead of ANSI-only, as before. This means that the virus can infect any files that can be opened.

The virus uses hashes instead of names, but the hashes are sorted alphabetically according to the strings they represent. The virus uses a reverse polynomial to calculate the hash (that magical '0xEDB88320' value). Since the hashes are sorted alphabetically, the export table needs to be parsed only once for all of the APIs. Each API address is placed on the stack for easy access, but because stacks move downwards in memory, the addresses end up in reverse order in memory. The virus does not check that the exports exist, relying instead on the Structured Exception Handler to deal with any problems that occur. Of course, the required APIs should always be present in the kernel, so no errors should occur anyway. The hash table is not terminated explicitly. Instead, the virus checks the low byte of each hash that has been calculated, and exits when a particular value is seen. The assumption is that each hash is unique and thus when a particular value (which corresponds to the last entry in the list) is seen, the list has ended. While this is true in the case of this virus, it might result in unexpected behaviour if other APIs are added, for which the low byte happens to match.

'RANDOM' NUMBER GENERATION

The virus calls the GetTickCount() function 16 times in a row, to initialize the state for the random number generator. This is a very poor way to seed the generator, given that if the host machine is reasonably modern, all of the numbers will be the same. The random number generator is WELL512 (see below), but the code is not a direct translation from the available C code. Instead, the constants have been adjusted to integrate the effect of some of the shifts. It is not known why this was done – the implementation is larger than can be achieved simply by translating the original version. One reason might be to hide the constants to make the algorithm more difficult to identify, but that seems pointless for a random number generator. It is far more common to do that for an encryption algorithm. It is also not known whether it was the virus author who produced this version of the code.

WELL, WELL, WELL

WELL512 is the 'Well Equidistributed Long-period Linear' random number generator – a smaller and faster random number generator than the Mersenne Twister, and written nearly ten years later. One implementation of WELL has an equal length of period to the Mersenne Twister, and the two generators even share a co-author. The main difference between WELL and the Mersenne Twister is that WELL improves on the Mersenne Twister by producing a 'better' distribution of values.

BITS AND PIECES

The virus searches in the current directory (only) for PE files, regardless of their extension. It uses a nice trick to find the files, that was first seen in the Chiton [1] family: the file mask is '*' which, when pushed onto the stack, can be interpreted as a zero-terminated Unicode string because it is followed by three zeroes. Another 'advancement' is that the virus attempts to remove the read-only attribute from whatever is found. This is in contrast to all of the previous viruses by the same author, which could not infect files if the read-only attribute was set. The virus attempts to open the found object and map a view of it. If the object is a directory, then this action will fail and the map pointer will be null. Any attempt to inspect such an object will cause an exception to occur, which the virus will intercept. If the map can be created, then the virus will inspect the file for its ability to be infected.

The virus is interested in Portable Executable files for the *Intel* x86 platform that are not DLLs or system files. The check for system files could serve as a light inoculation method, since *Windows* ignores this flag. The virus checks the COFF magic number, which is unusual, but correct. The reason for checking the value of the COFF magic number is to be sure that the file is a 32-bit image. This is the safest way to determine that fact because, apart from the 'IMAGE_FILE_EXECUTABLE_IMAGE' and 'IMAGE_FILE_DLL' flags in the Characteristics field, all of the other flags are ignored by *Windows*. This includes the flag ('IMAGE_FILE_32BIT_MACHINE') that specifies that the file is for 32-bit systems. As an added precaution, the virus checks for the size of the optional header being the standard value.

The virus also requires that the file has no Load Configuration Table, because the table includes the SafeSEH structures, which will prevent the virus from using arbitrary exceptions to transfer control to other locations within its body. The last two checks that the virus performs are that the file targets the GUI subsystem, and that it has a Base Relocation Table which begins at exactly the start of the last section, and which is at least as large as the virus body.

ADVANCED ENCRYPTION SOMETIMES

The virus constructs a decryptor that uses the AES instruction set to decrypt itself. The decryptor begins by pushing the host entry point RVA onto the stack, and then calling the decryption routine. The virus does not check if the CPU supports the AES instructions because the assumption is that the exception handler that the virus registers will intercept any errors. There is a minor problem with this assumption, which is that the host file has been altered irrevocably at this point – the relocation table data has partially been overwritten. However, the file will not be considered by *Windows* to be corrupted – it will continue to load as before, but it might not run properly (see below). The file will also remain a candidate for infection if it is transferred to a platform where the AES instructions are supported. The first generation of the virus carries the value '1986' repeatedly in the place where the AES key will be stored. This seems likely to be a reference to the virus writer's handle, and might be her birth year.

The virus calculates four random numbers and stores them in the virus body (technically, they are four parts of a single 16-byte number). The virus also pushes the last of those four numbers onto the stack. There is a funny piece of code at this point – the virus explicitly caches the stack pointer in a register and then uses the 'enter' instruction, which implicitly caches the stack pointer in the ebp register. Later, the original stack pointer is restored from the cache register. This is particularly surprising, given that the 'leave' instruction exists for that exact purpose, and the virus author has made correct use of the 'enter' instruction in another virus [2].

The virus copies the 16-byte number onto the stack. It calls a routine which executes the 'aeskeygenassist' instruction on the 16-byte number and then performs various bit-shufflings and XORs on the result, which is also saved on the stack for use later. This routine is called separately once, and then seven more times in a loop. The round constant begins with 1, and is shifted once per iteration of the loop. It is not known why the virus author didn't rotate the value instead of shifting it, which would have allowed the routine to be called eight times in a loop, thus avoiding the initial call. The routine is then called twice more, once with a round constant of 27 and once with a round constant of 54. It is also not known why these numbers were chosen.

The result is a total of 11 random numbers, each of which is 16 bytes in length. The first random number is XORed against 16 bytes of the virus body, and then the next ten random numbers are used to encrypt the result. There is an additional encryption operation using two registers that are not initialized and have no effect on the result. This do-nothing operation is actually a place holder for a decryption operation in the decryptor. The instruction is replaced with an 'aesimc' instruction in the decryptor, which is used to prepare the round key for decryption. It is also interesting to note that the act of single-stepping through the code using the *WinDbg* debugger causes all of the XMM registers to be zeroed, completely breaking the encryption operations. Finally, the values on the stack are discarded by restoring the stack pointer using the cache register. The stack is ultimately balanced by discarding the random number that was pushed originally, and which served no purpose at all.

TOUCH AND GO

The virus overwrites the relocation table with the encrypted virus body, changes the section characteristics to writable and executable, and sets the host entry point to point directly to the virus code. The virus clears only two flags in the DLL Characteristics field: IMAGE_DLLCHARACTERISTICS_FORCE_INTEGRITY and IMAGE_DLLCHARACTERISTICS_NO_SEH. This allows signed files to be altered without triggering an error, and enables Structured Exception Handling. The virus also zeroes the Base Relocation Table data directory entry. This is probably intended to disable Address Space Layout Randomization (ASLR) for the host, but it also serves as the infection marker. Unfortunately for the virus writer, this has no effect at all against ASLR. The 'problem' is that ASLR does not require relocation data for a process to be 'relocated'. If the file specifies that it supports ASLR, then it will always be loaded to a random address. The only difference between the presence and absence of relocation data is that without it, no content in the process will be altered. *Windows* assumes that if the process specifies that it supports ASLR, then it really does support ASLR, no matter what the structure of the file looks like. The result is that a process that had a relocation table overwritten by the virus will crash when it attempts to access its variables using the original unrelocated addresses. Alternatively, if the platform does not support ASLR (i.e. *Windows XP* and earlier), and if something else is already present at the host load address (or if the load address is intentionally invalid to force the use of the relocation table), then the file will no longer load.

After the infection is complete, the virus unmaps the view and then attempts to close the handle, but there is a bug at this point (technically, there are two bugs of the same kind): the wrong offset is used when indexing into the structure that holds the API addresses. Instead of calling the CloseHandle() function, the virus calls the UnmapViewOfFile() function again. Twice, in fact: once for each of the handles that is supposed to be closed. As a result, the virus leaks one handle for every file that is opened successfully, and an additional handle for every file that is mapped successfully.

DECRYPTOR

The decryptor begins by caching the stack pointer in a register and then rounding down the stack pointer to the nearest multiple of 64KB, before saving the extended floating point state onto the stack. The reason for aligning the stack pointer is that the 'fxsave' instruction requires that the destination address is aligned to a multiple of 16 bytes. However, the state is 512 bytes long, so the virus makes sure that saving the state will not corrupt the variables that are already on the stack. The virus decrypts itself by executing the same encryption instructions as before, but in reverse order. After replication is complete, the virus restores the extended floating point state and returns the stack pointer to its original value before passing control to the host.

CONCLUSION

The use of the AES instruction set will certainly challenge CPU emulators in the short term, but the lack of a preceding CPUID check for its support might be considered quite suspicious for now (in the same way that almost no one checks for the presence of support for the MMX instruction set before attempting to use it). For now, at least, time is on our side.

REFERENCES

[1] http://www.virusbtn.com/pdf/magazine/2002/200206.pdf.

[2] http://www.virusbtn.com/pdf/magazine/2012/201206.pdf.

CALL FOR PAPERS

VB2013 BERLIN

Virus Bulletin is seeking submissions from those wishing to present papers at VB2013, which will take place 2–4 October 2013 at the Maritim Hotel Berlin, Germany.

The conference will include a programme of 30-minute presentations running in two concurrent streams: Technical and Corporate.

Submissions are invited on all subjects relevant to anti-malware and anti-spam. In particular, *VB* welcomes the submission of papers that will provide delegates with ideas, advice and/or practical techniques, and encourages presentations that include practical demonstrations of techniques or new technologies.

A list of topics suggested by the attendees of VB2012 can be found at http://www.virusbtn.com/conference/vb2013/call/. However, please note that this list is not exhaustive, and the selection committee will consider papers on these and any subjects relevant to the anti-malware, anti-spam and related security communities.

SUBMITTING A PROPOSAL

The deadline for submission of proposals is **Friday 8 March 2013**. Abstracts should be submitted via our online abstract submission system. You will need to include:

- An abstract of approximately 200 words outlining the proposed paper and including five key points that you intend the paper to cover.
- Full contact details.
- An indication of whether the paper is intended for the Technical or Corporate stream.

The abstract submission form can be found at http://www.virusbtn.com/conference/abstracts/.

One presenter per selected paper will be offered a complimentary conference registration, while co-authors will be offered registration at a 50% reduced rate (up to a maximum of two co-authors). *VB* regrets that it is not able to assist with speakers' travel and accommodation costs.

Authors are advised that, should their paper be selected for the conference programme, they will be expected to provide a full paper for inclusion in the VB2013 Conference Proceedings as well as a 30-minute presentation at VB2013. The deadline for submission of the completed papers will be 10 June 2013, and potential speakers must be available to present their papers in Berlin between 2 and 4 October 2013.

Any queries should be addressed to editor@virusbtn.com.

MALWARE ANALYSIS 2

PLEASE HELP!

Raul Alvarez
Fortinet, Canada

These days, we seldom use the help (.hlp) file in our daily computing tasks. We have the Internet at our disposal for more or less everything that we need to learn. What's more, .hlp files are not supported by all operating systems.

In this article, we will discuss code that drops malware onto your local machine once an .hlp file is opened. We will look into the execution path of a piece of malware that resembles a piece of shellcode inside an .hlp file.

OVERVIEW

Recently, researchers found a help file named 'Amministrazione.hlp' that installs a keylogger on the local machine. Once the file is opened, it shows a message stating that the file could not be read. In the background, however, a file is being dropped, which in turn creates the keylogger. We have also seen another .hlp file which has similar code that runs in the background. It is safe to assume that the two samples belong to the same family. We will investigate these pieces of code as we move on.

SAMPLE #1

Let's look at the first sample.

The malware first decrypts 716 bytes using a simple decryption algorithm, as shown below. It uses the basic assembly instructions: SUB (subtract), SHL (Shift Left), and ADD. LODS is used to grab the WORD value from the current ESI and store it at the AX register, while the STOS instruction is used to store the value of the AL register in the memory pointed to by EDI:

```
LODS WORD PTR DS:[ESI]
SUB AX,6161
SHL AL,4
ADD AL,AH
STOS BYTE PTR ES:[EDI]
```

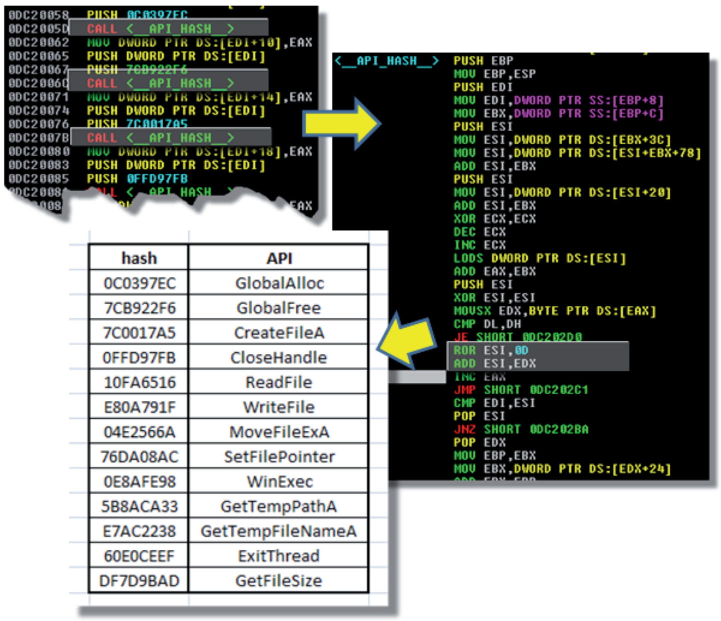
API resolution

After the decryption routine, the malware parses the PEB (Process Environment Block) to locate the imagebase of kernel32.dll. Once the imagebase is acquired, it locates the export table of kernel32.dll and searches for the first API names in the table. The malware acquires the addresses of all the APIs it requires by hashing the API names found in the kernel32.dll export table and comparing them to its own list of values.

The malware has a list of constant values that are equivalent to the hashes of the API names that it needs (see Figure 1). These constant values are the input parameters for the subroutine that returns the addresses of the APIs.

The subroutine first computes the hash value of the first API name found in the export table of kernel32, then compares it with the current constant hash value. If the two are not the same, it will proceed to the next API name in the table. The computation of the hash value for each API name will continue until it finds one that matches the current constant hash value.

The API names are hashed using simple ROR and ADD instructions. (Figure 1 shows a snapshot of the code used for hashing and the table of API hashes used by the malware.) Once the correct hash has been found, the address of that API becomes the resulting value returned by the subroutine.



```
0DC20058  PUSH 0C0397EC
0DC2005D  CALL <__API_HASH__>
0DC20062  MOV DWORD PTR DS:[EDI+10],EAX
0DC20065  PUSH DWORD PTR DS:[EDI]
0DC20067  PUSH 7CB922F6
0DC2006C  CALL <__API_HASH__>
0DC20071  MOV DWORD PTR DS:[EDI+14],EAX
0DC20074  PUSH DWORD PTR DS:[EDI]
0DC20076  PUSH 7C0017A5
0DC2007B  CALL <__API_HASH__>
0DC20080  MOV DWORD PTR DS:[EDI+18],EAX
0DC20083  PUSH DWORD PTR DS:[EDI]
0DC20085  PUSH 0FFD97FB
```

```
<__API_HASH__>  PUSH EBP
                MOV EBP,ESP
                PUSH EDI
                MOV EDI,DWORD PTR SS:[EBP+8]
                MOV EBX,DWORD PTR SS:[EBP+C]
                PUSH ESI
                MOV ESI,DWORD PTR DS:[EBX+3C]
                MOV ESI,DWORD PTR DS:[ESI+EBX+78]
                ADD ESI,EBX
                PUSH ESI
                MOV ESI,DWORD PTR DS:[ESI+20]
                ADD ESI,EBX
                XOR ECX,ECX
                DEC ECX
                INC ECX
                LODS DWORD PTR DS:[ESI]
                ADD EAX,EBX
                PUSH ESI
                XOR ESI,ESI
                MOVSX EDX,BYTE PTR DS:[EAX]
                CMP DL,DH
                JE SHORT 0DC202D0
                ROR ESI,0D
                ADD ESI,EDX
                INC EAX
                JMP SHORT 0DC202C1
                CMP EDI,ESI
                POP ESI
                JNZ SHORT 0DC202BA
                POP EDX
                MOV EBP,EBX
                MOV EBX,DWORD PTR DS:[EDX+24]
```

hash	API
0C0397EC	GlobalAlloc
7CB922F6	GlobalFree
7C0017A5	CreateFileA
0FFD97FB	CloseHandle
10FA6516	ReadFile
E80A791F	WriteFile
04E2566A	MoveFileExA
76DA08AC	SetFilePointer
0E8AFE98	WinExec
5B8ACA33	GetTempPathA
E7AC2238	GetTempFileNameA
60E0CEEF	ExitThread
DF7D9BAD	GetFileSize

Figure 1: Snapshot of the code used for hashing and table of API hash values used by the malware.

Blind parsing

After getting all the required API addresses, the malware tries to locate the handle of the original help file by getting its file size.

The malware uses the GetFileSize API and a HANDLE parameter with a value of 1. If the result is INVALID_ FILE_SIZE (0xffffffff), it will increase the HANDLE value by 1 and check the file size again. It will keep increasing the HANDLE value until it gets a result other than INVALID_FILE_SIZE. Once the result of GetFileSize is anything other than 0xffffffff, the malware assumes that the HANDLE is for a valid file.

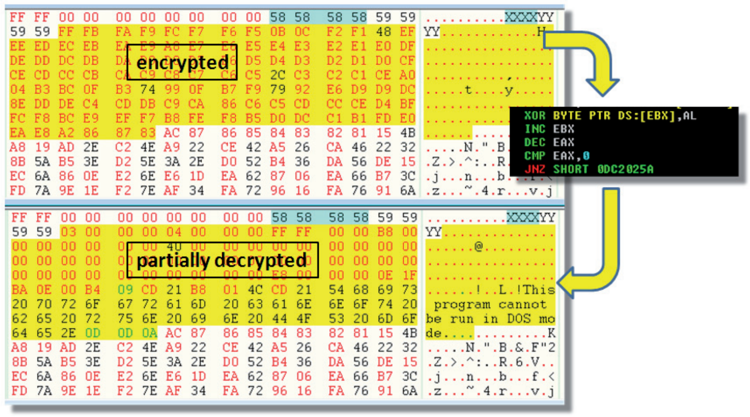
The malware reads and parses the contents of the file in memory by checking for 'XXXXYYYY'. If the marker is not found, it will go back to the loop that gets the file size of the incremented HANDLE. If 'XXXXYYYY' is found, the malware double checks that it is reading the right file by checking for the second marker, 'YYYYXXXX'.

Dropped file

Once the correct help file is loaded in memory, it creates an empty TMP file in a %temp% folder using the GetTempPathA and GetTempFileNameA APIs. It also sets the newly created TMP file to be deleted on the next reboot by calling the MoveFileExA API and sets the parameters NewName to NULL and Flags to DELAY_UNTIL_REBOOT.

The malware decrypts the rest of the malware body, which is loaded in memory, using a simple XOR instruction with decrementing key values. It writes the decrypted malware to the TMP file (see Figure 2).

The malware transfers control to the TMP file by calling the WinExec API.



```
FF FF 00 00  00 00 00 00  00 00 58 58  58 58 59 59  ..........XXXXYY
59 59 FF FB  FA F9 FC F7  F6 F5 0B 0C  F2 F1 48 EF  YY............H.
EE ED EC EB  EA E9 A8 E7  E6 E5 E4 E3  E2 E1 E0 DF  ................
DE DD DC DB  DA [illegible]  D5 D4 D3  D2 D1 D0 CF  ................
CE CD CC CB  CA C9 C8 C7  C6 C5 2C C3  C2 C1 CE A0  ..........,.....
04 B3 BC 0F  B3 74 99 0F  B7 F9 79 92  E6 D9 D9 DC  .....t....y.....
8E DD DE C4  CD DB C9 CA  86 C6 C5 CD  CC CE D4 BF  ................
FC F8 BC E9  EF F7 B8 FE  F8 B5 D0 DC  C1 B1 FD E0  ...........
EA E8 A2 86  87 83 AC 87  86 85 84 83  82 81 15 4B  ...........
A8 19 AD 2E  C2 4E A9 22  CE 42 A5 26  CA 46 22 32  .....N.".B
8B 5A B5 3E  D2 5E 3A 2E  D0 52 B4 36  DA 56 DE 15  .Z.>.^:..R
EC 6A 86 0E  E2 6E E6 1D  EA 62 87 06  EA 66 B7 3C  .j...n...b
FD 7A 9E 1E  F2 7E AF 34  FA 72 96 16  FA 76 91 6A  .z...~.4.r...v.j
```

```
XOR BYTE PTR DS:[EBX],AL
INC EBX
DEC EAX
CMP EAX,0
JNZ SHORT 0DC2025A
```

```
FF FF 00 00  00 00 00 00  00 00 58 58  58 58 59 59  ..........XXXXYY
59 59 03 00  00 00 04 00  00 00 FF FF  00 00 B8 00  YY..............
00 00 00 00  00 00 40 00  00 00 00 00  00 00 00 00  ......@.........
00 00 00 00  0[illegible]  00 00 00 00  ................
00 00 00 00  00 00 00 00  00 00 E8 00  00 00 0E 1F  ................
BA 0E 00 B4  09 CD 21 B8  01 4C CD 21  54 68 69 73  ......!..L.!This
20 70 72 6F  67 72 61 6D  20 63 61 6E  6E 6F 74 20   program cannot 
62 65 20 72  75 6E 20 69  6E 20 44 4F  53 20 6D 6F  be run in DOS mo
64 65 2E 0D  0D 0A AC 87  86 85 84 83  82 81 15 4B  de.............K
A8 19 AD 2E  C2 4E A9 22  CE 42 A5 26  CA 46 22 32  .....N.".B.&.F"2
8B 5A B5 3E  D2 5E 3A 2E  D0 52 B4 36  DA 56 DE 15  .Z.>.^:..R.6.V..
EC 6A 86 0E  E2 6E E6 1D  EA 62 87 06  EA 66 B7 3C  .j...n...b...f.<
FD 7A 9E 1E  F2 7E AF 34  FA 72 96 16  FA 76 91 6A  .z...~.4.r...v.j
```

Figure 2: The decrypted malware is written to the TMP file.

Once the TMP file executes, it drops the following files in the '\Documents and Settings\[username]\Local Settings\Application Data\' folder:

```
RECYCLER.dll
Windows Security Center.exe
UserData.dat
Windows Security Center.lnk
```

SAMPLE #2

This sample also starts by decrypting approximately 738 bytes of code/data. It uses a different decryption algorithm from the first sample, using a combination of INC (Increment), IMUL (signed multiplication), and XOR instructions:

```
INC ECX
INC ECX
INC EDX
IMUL EAX,DWORD PTR DS:[ECX+41],10
XOR AL,BYTE PTR DS:[ECX+42]
XOR AL,BYTE PTR DS:[EDX+42]
XOR BYTE PTR DS:[EDX+42],AL
POP EAX
PUSH EAX
CMP BYTE PTR DS:[ECX+43],AL
```

API resolution

The API resolution is almost the same as for the first sample. The malware acquires the imagebase of kernel32.dll by parsing the PEB, and also locates the export table to hash the API names.

It is interesting to note that the first sample started by getting the hash value of the first API name found in the kernel32 export table, but sample 2 starts from the last API name found in the table.

Sample 2 uses the same computation to hash the required APIs. Although the list of required APIs is not exactly the same as for sample 1, as shown in Figure 3, the hash values themselves are a strong indicator that the two samples belong to the same family.

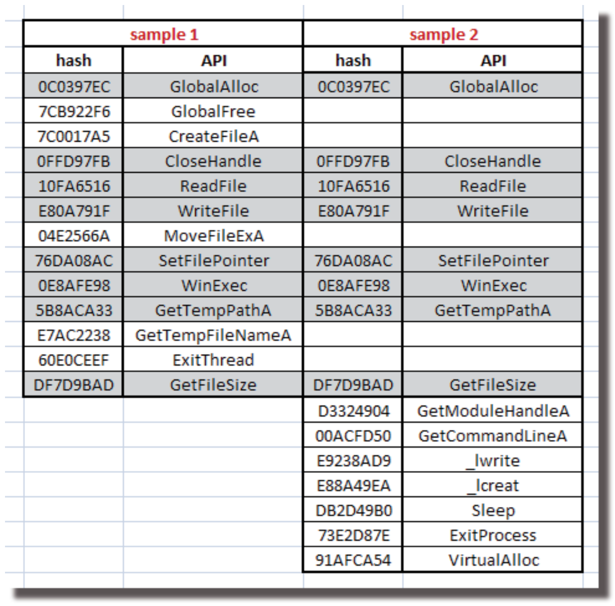
Blind parsing

Sample 2 uses the same technique as used by sample 1 to locate the original help file. The file marker for this sample is 57 64 50 49 EF FE EB BE (in hex). Once this marker is found, the malware knows that it is looking into the original help file.

Dropped file

Sample 2 decrypts the rest of the malware body but does not drop a separate file. Instead, after decrypting the rest of the malware and putting it into the allocated memory, it transfers control to the newly decrypted code.

We will not follow the execution of the newly decrypted code, since we are looking at the execution of the common code between samples 1 and 2.



sample 1		sample 2	
hash	API	hash	API
0C0397EC	GlobalAlloc	0C0397EC	GlobalAlloc
7CB922F6	GlobalFree		
7C0017A5	CreateFileA		
0FFD97FB	CloseHandle	0FFD97FB	CloseHandle
10FA6516	ReadFile	10FA6516	ReadFile
E80A791F	WriteFile	E80A791F	WriteFile
04E2566A	MoveFileExA		
76DA08AC	SetFilePointer	76DA08AC	SetFilePointer
0E8AFE98	WinExec	0E8AFE98	WinExec
5B8ACA33	GetTempPathA	5B8ACA33	GetTempPathA
E7AC2238	GetTempFileNameA		
60E0CEEF	ExitThread		
DF7D9BAD	GetFileSize	DF7D9BAD	GetFileSize
		D3324904	GetModuleHandleA
		00ACFD50	GetCommandLineA
		E9238AD9	_lwrite
		E88A49EA	_lcreat
		DB2D49B0	Sleep
		73E2D87E	ExitProcess
		91AFCA54	VirtualAlloc

Figure 3: Hash values and APIs.

After executing the newly decrypted code, the malware creates a new file in the %temp% folder named 'help.hlp', using the _lcreat and _lwrite APIs.

Finally, the malware uses the WinExec API to open the help.hlp file, using 'cmd.exe /c %temp%\help.hlp' as the parameter. 'Help.hlp' contains the actual help document that displays something that looks like a news article.

CONCLUSION

In previous articles we have discussed Quervar [1], which infects document files, and which can also infect other files that contain the document extension name. We have also looked at the DLL-infecting Floxif [2]. Now, we are looking at infected help files. It seems as if pretty much any type of file that exists on our computer is susceptible to infection. The most reliable form of defence is to make sure anti-malware software is kept up to date and that our computers are scanned regularly. Stay safe!

REFERENCES

[1] Alvarez, R. Filename: BUGGY.COD.E. Virus Bulletin, October 2012. http://www.virusbtn.com/virusbulletin/archive/2012/10/vb201210-Quervar.

[2] Alvarez, R. Compromised Library. Virus Bulletin, December 2012. http://www.virusbtn.com/virusbulletin/archive/2012/12/vb201212-Floxif.

FEATURE

TECHNIQUES FOR EVADING AUTOMATED ANALYSIS

Abhishek Singh
FireEye, USA

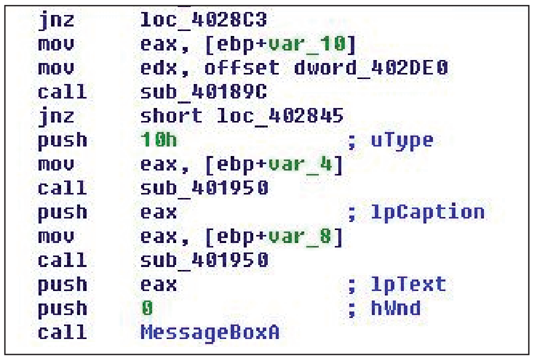
Automated analysis is widely used by the anti-virus industry in order to process the enormous number of new malware samples that appear every day. In order to evade detection by such automated analysis, malware authors employ various techniques to detect virtual environments, or sandboxes, so that they can bypass them. This paper presents some of the techniques that are commonly used to bypass analysis in a virtualized environment. The techniques have been classified into three categories: evasion techniques based on a lack of human interaction, evasion techniques employing environment-specific checks, and evasion techniques employing behavioural-specific checks.

HUMAN INTERACTION

A virtualized environment is devoid of human interaction such as the clicking of mouse buttons and use of the keyboard. This fact is used by many malicious programs to evade automated analysis.

Using message boxes for activation

Many exploit tools, such as LOIC and the BOMBA exploit pack, evade detection in a sandbox by employing message boxes for activation.



```
jnz     loc_4028C3
mov     eax, [ebp+var_10]
mov     edx, offset dword_402DE0
call    sub_40189C
jnz     short loc_402845
push    10h             ; uType
mov     eax, [ebp+var_4]
call    sub_401950
push    eax             ; lpCaption
mov     eax, [ebp+var_8]
call    sub_401950
push    eax             ; lpText
push    0               ; hWnd
call    MessageBoxA
```

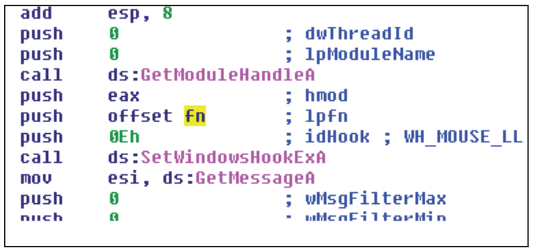
Figure 1: Malware using MessageBox for activation.

The MessageBox and MessageBoxEx APIs display dialog boxes, sets of buttons and specific messages. As shown in Figure 1, the function returns an integer value indicating that a button has been clicked (and which one). The rest of the code will only execute after this integer value has been returned (i.e. after a response has been registered).

In the case of an automated analysis system, since there is no human interaction (no buttons being clicked), the malicious code will remain dormant. However, in a real-life scenario, a user is likely to click the button, thus activating the malware.

Hooking mouse activity

Many pieces of malware, such as Trojan UpClicker, employ hooking of the mouse to evade sandbox analysis. As shown in Figure 2, UpClicker calls the SetWindowsHookExA() function with 0Eh as a parameter. This is used to hook the mouse. Function fn is the pointer to the callback procedure in the code, i.e. when mouse activity takes place, the code pointed to by the function fn will be called.

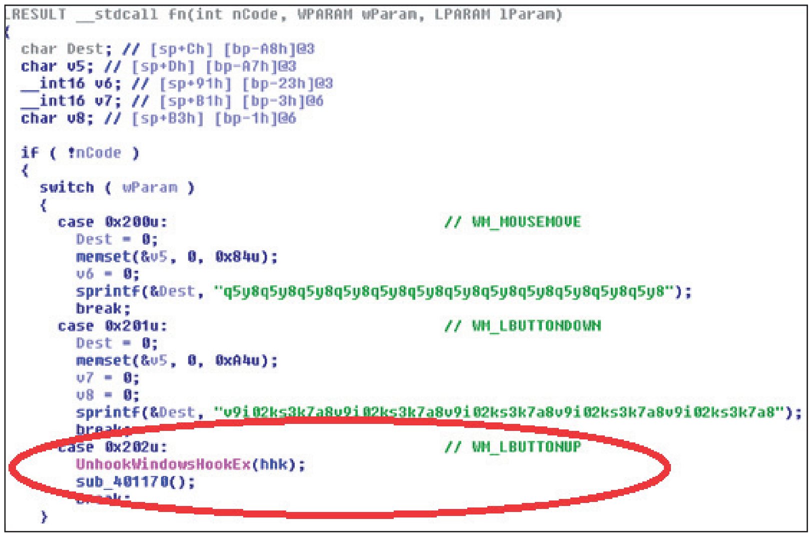


```
add     esp, 8
push    0                ; dwThreadId
push    0                ; lpModuleName
call    ds:GetModuleHandleA
push    eax              ; hmod
push    offset fn        ; lpfn
push    0Eh              ; idHook ; WH_MOUSE_LL
call    ds:SetWindowsHookExA
mov     esi, ds:GetMessageA
push    0                ; wMsgFilterMax
push    0                ; wMsgFilterMin
```

Figure 2: UpClicker code showing hooking of the mouse.

The code pointed to by fn (see Figure 3) reveals that the function monitors mouse movements. Specifically, it monitors the press of the left mouse button down and up.

As can be seen from the code shown in Figure 3, the UnhookWindowsHookEx() function is only called when the left mouse button moves up. The call to this function will unhook the malicious code from the mouse, after which it makes a call to the sub_401170() function, which executes the malicious code. So until the left mouse button is released (which won't happen in an automated analysis system as there is no human interaction/mouse activity), the code will remain dormant, making it immune from automated analysis in a sandbox.



```
LRESULT __stdcall fn(int nCode, WPARAM wParam, LPARAM lParam)
{
  char Dest; // [sp+Ch] [bp-A8h]@3
  char v5; // [sp+Dh] [bp-A7h]@3
  __int16 v6; // [sp+91h] [bp-23h]@3
  __int16 v7; // [sp+81h] [bp-3h]@6
  char v8; // [sp+B3h] [bp-1h]@6

  if ( !nCode )
  {
    switch ( wParam )
    {
      case 0x200u:                                    // WM_MOUSEMOVE
        Dest = 0;
        memset(&v5, 0, 0x84u);
        v6 = 0;
        sprintf(&Dest, "q5y8q5y8q5y8q5y8q5y8q5y8q5y8q5y8q5y8q5y8q5y8");
        break;
      case 0x201u:                                    // WM_LBUTTONDOWN
        Dest = 0;
        memset(&v5, 0, 0xA4u);
        v7 = 0;
        v8 = 0;
        sprintf(&Dest, "v9i02ks3k7a8v9i02ks3k7a8v9i02ks3k7a8v9i02ks3k7a8v9i02ks3k7a8");
        break;
      case 0x202u:                                    // WM_LBUTTONUP
        UnhookWindowsHookEx(hhk);
        sub_401170();
        break;
    }
```

Figure 3: Code pointed to by fn.

ENVIRONMENT-SPECIFIC EVASION

An automated analysis system may make use of *VMware*, *VPC* or *VirtualBox* to create an isolated environment. These environments will have processes, services and product keys that are specific to the environment (i.e. specific to *VMware*, *VPC* or *VirtualBox*). These specific files, processes and product keys are searched for by the malicious code, and it will not execute if any of these indicators are spotted.

Enumerating the system service list specific to VMware

VMware is widely used for creating a virtualized environment for automated analysis. In order to detect the presence of *VMware*, one technique employed by malware is to check for specific services used only by *VMware*. Some of these are: vmicheatbeat, vmci, vmdebug, vmmouse, vmscis, VMTools, vmware, vmx86, vmhgfs and vmxnet.

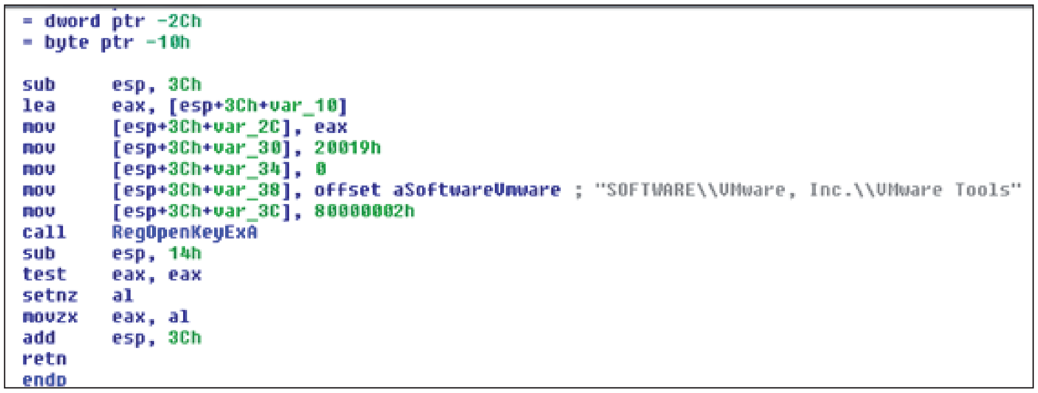
One of the methods for detecting such services is to use the RegOpenKeyExA() function and check for the services (see Figure 4). If the RegOpenKeyExA() function succeeds, the return value will be a non-zero error code.

Another way to detect the presence of *VMware* is to check for the presence of files used by the program. For example, malware may use the GetFileAttributeA() function with the files used by *VMware* as its parameter (see Figure 5).

The GetFileAttributeA() function will retrieve the system attributes for a specified file or directory. As shown in Figure 5, after the function call, the code 'cmp eax, 0FFFFFFFFh' checks if the value returned is -1. This denotes that the function is unable to retrieve the attributes of the file (in this case vmmouse.sys), hence in this case the environment in which the code is executing is not a *VMware* virtualized environment.

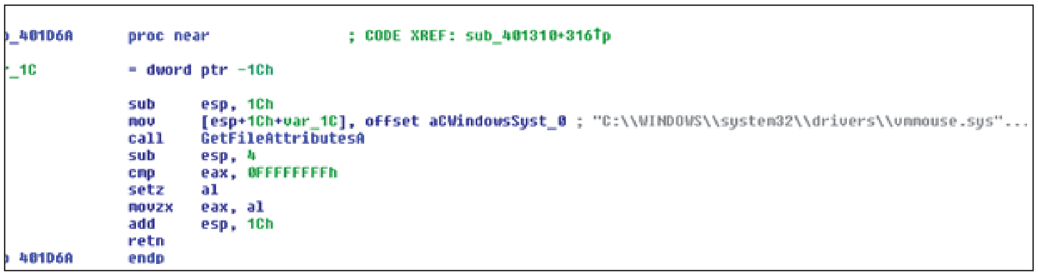
Checking the communication port

VMware uses the VMX port to communicate with the virtual machine. The port is checked by malware to detect the presence of *VMware*. The flow of instructions is shown in Figure 6.



```
= dword ptr -2Ch
= byte ptr -10h

sub     esp, 3Ch
lea     eax, [esp+3Ch+var_10]
mov     [esp+3Ch+var_2C], eax
mov     [esp+3Ch+var_30], 20019h
mov     [esp+3Ch+var_34], 0
mov     [esp+3Ch+var_38], offset aSoftwareVmware ; "SOFTWARE\\VMware, Inc.\\VMware Tools"
mov     [esp+3Ch+var_3C], 80000002h
call    RegOpenKeyExA
sub     esp, 14h
test    eax, eax
setnz   al
movzx   eax, al
add     esp, 3Ch
retn
endp
```

Figure 4: Malware using the RegOpenKeyExA() function to check for the presence of VMware.



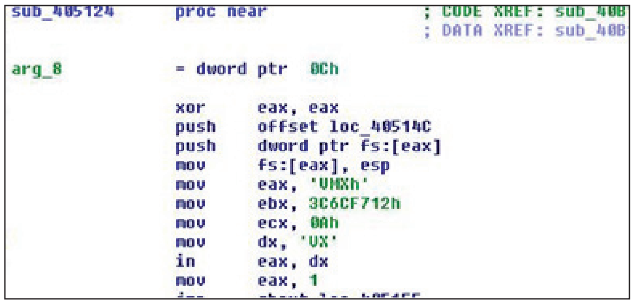
```
_401D6A      proc near               ; CODE XREF: sub_401310+316↑p

_1C          = dword ptr -1Ch

             sub     esp, 1Ch
             mov     [esp+1Ch+var_1C], offset aCWindowsSyst_0 ; "C:\\WINDOWS\\system32\\drivers\\vmmouse.sys"...
             call    GetFileAttributesA
             sub     esp, 4
             cmp     eax, 0FFFFFFFFh
             setz    al
             movzx   eax, al
             add     esp, 1Ch
             retn
 401D6A      endp
```

Figure 5: Malware using GetFileAttributeA() to determine the presence of vmmouse.

The instruction 'move eax, "VMXh"' loads the value 0x564D5868 into the EAX register. In the subsequent instruction, EBX is loaded with any value, after which ECX is set to 0Ah. This value (0Ah) will get the *VMware* version. The DX register is set to the port 'VX' which, as discussed above, allows interfacing with *VMware*. The instruction 'in eax, dx' is then called to read from the port into EAX. In the presence of *VMware* the call will succeed, while if *VMware* is absent an exception will occur. Once a malicious program has detected the presence of *VMware*, it will stop execution.

Checking for product ID keys

Besides checking the execution environment for *VirtualBox*, *VMware* and *VPC* images, many malicious programs also check for proprietary automated malware analysis systems such as *Sandboxie* or *Joe Sandbox*, *Anubis* and *CWSandbox*. One of the commonly used methods to check for the presence of these is to check for a unique product key.



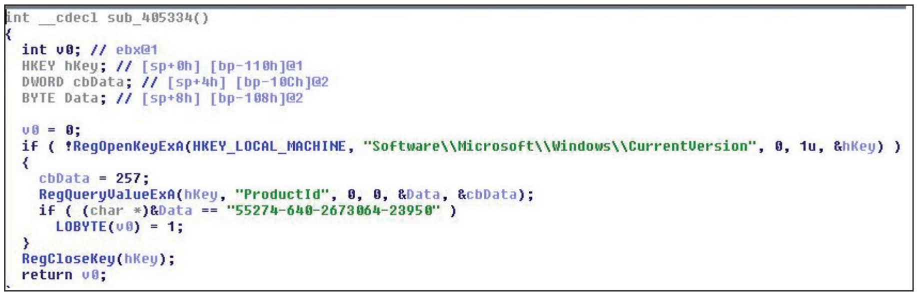
```
sub_405124      proc near                       ; CODE XREF: sub_40B
                                                ; DATA XREF: sub_40B

arg_8           = dword ptr  0Ch

                xor     eax, eax
                push    offset loc_40514C
                push    dword ptr fs:[eax]
                mov     fs:[eax], esp
                mov     eax, 'VMXh'
                mov     ebx, 3C6CF712h
                mov     ecx, 0Ah
                mov     dx, 'VX'
                in      eax, dx
                mov     eax, 1
```

Figure 6: Malware using IO ports to detect VMware.

As shown in Figure 7, to check the product ID, the registry key HKLM\Software\Microsoft\Windows\CurrentVersion\ is opened using the RegOpenKeyExA() function. After that, a call is made to the RegQueryValueEx() function. The RegQueryValueEx() function retrieves the type and data for the specified value associated with the open registry key. The data value returned by the function is then compared with known product IDs. If the value matches the product ID for *Joe Sandbox*, *CWSandbox* or the *Anubis* sandbox, the presence of these automated analysis systems can be detected.

Checking for processes/DLLs specific to proprietary automated analysis systems

Another way to detect proprietary automated analysis systems is to check for the presence of processes and DLLs that are specific to the systems. As shown in Figure 8, the malware uses the GetModuleHandleA() function to get the handle of the file 'sbiedl.dll'. This is a DLL that is loaded within the sbiectrl.exe process of *Sandboxie*, a proprietary sandbox used for automated analysis.



```
int __cdecl sub_405334()
{
  int v0; // ebx@1
  HKEY hKey; // [sp+0h] [bp-110h]@1
  DWORD cbData; // [sp+4h] [bp-10Ch]@2
  BYTE Data; // [sp+8h] [bp-108h]@2

  v0 = 0;
  if ( !RegOpenKeyExA(HKEY_LOCAL_MACHINE, "Software\\Microsoft\\Windows\\CurrentVersion", 0, 1u, &hKey) )
  {
    cbData = 257;
    RegQueryValueExA(hKey, "ProductId", 0, 0, &Data, &cbData);
    if ( (char *)&Data == "55274-640-2673064-23950" )
      LOBYTE(v0) = 1;
  }
  RegCloseKey(hKey);
  return v0;
```

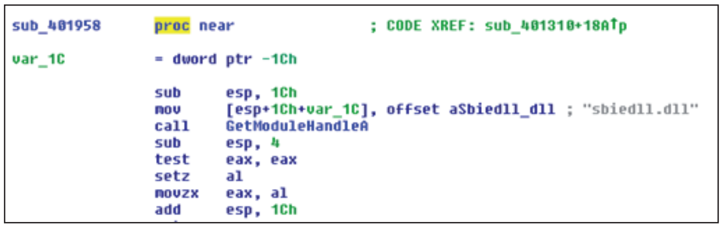
Figure 7: Malware checking specific product ID.

If the GetModuleHandleA() function is able to get the handle of the sbiedl.dll module, then the malicious code will know that it is executing inside *Sandboxie*.

As shown in Figure 9, malware checks for the presence of *Joe Sandbox* by checking for processes specific to the system including joeboxserver.exe and joeboxcontrol.exe. If any of the processes are present, the malware will not execute.

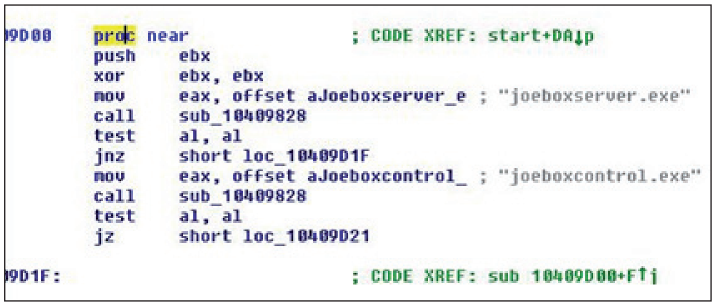
CONFIGURATION-SPECIFIC EVASION

Automated analysis systems are configured with pre-defined parameters. For example, the system will execute a sample for a specified time. It will have a pre-defined version of an application installed in the environment. Configuration-specific evasion makes use of these configuration issues to bypass automated analysis systems.



```
sub_401958      proc near               ; CODE XREF: sub_401310+18A↑p

var_1C          = dword ptr -1Ch

                sub     esp, 1Ch
                mov     [esp+1Ch+var_1C], offset aSbiedll_dll ; "sbiedll.dll"
                call    GetModuleHandleA
                sub     esp, 4
                test    eax, eax
                setz    al
                movzx   eax, al
                add     esp, 1Ch
```

Figure 8: Malware checking for the presence of Sandboxie.

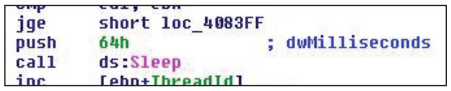


```
9D00      proc near                   ; CODE XREF: start+DA↓p
          push    ebx
          xor     ebx, ebx
          mov     eax, offset aJoeboxserver_e ; "joeboxserver.exe"
          call    sub_10409828
          test    al, al
          jnz     short loc_10409D1F
          mov     eax, offset aJoeboxcontrol_ ; "joeboxcontrol.exe"
          call    sub_10409828
          test    al, al
          jz      short loc_10409D21

9D1F:                                 ; CODE XREF: sub 10409D00+F↑j
```

Figure 9: Malware checking for Joe Sandbox.

Using sleep calls to delay execution

Virtualized environments are set to execute malware within a given time frame. One way for malware to evade automatic analysis is to use a sleep call with a long delay. As shown in Figure 10, sleep calls accept parameters in milliseconds. The execution of the code is suspended for a number of milliseconds.



```
jge     short loc_4083FF
push    64h                 ; dwMilliseconds
call    ds:Sleep
inc     [ebp+ThreadId]
```

Figure 10: Malware using sleep calls.

Since the virtualized environment is set to capture the behaviour of a piece of malware within a given time frame, a long sleep call will delay the execution of the malware, thus bypassing analysis by the automated system.

CONCLUSION

Automated analysis is a key component for malware research. It saves time and enables enormous numbers of samples to be examined and processed. Besides having the code to exploit a system, many pieces of malware also carry code to evade automated analysis. This may take advantage of methods which require human interaction, employ methods that locate the product keys, specific files, specific DLLs or processes spawned by the automated analysis systems, or it may make use of pre-defined configuration settings to bypass automated analysis. In the future we expect to see more samples that evade automated analysis, along with more techniques for doing so.

ACKNOWLEDGEMENT

The author would like to thank to Zheng Bu for providing invaluable feedback and guidance.

REFERENCES

[1] Omella, A.A. http://charette.no-ip.com:81/programming/2009-12-30_Virtualization/www.s21sec.com_vmware-eng.pdf.

[2] Mushtaq, A. http://blog.fireeye.com/research/2011/01/the-dead-giveaways-of-vm-aware-malware.html.

[3] Singh, A.; Khalid, Y. http://blog.fireeye.com/research/2012/12/dont-click-the-left-mouse-button-trojan-upclicker.html.

COMMENTARY 1

ANTI-VIRUS: LAST RITES, OR RITES OF PASSAGE?

David Harley
ESET, UK

Anti-virus is dead. Again. Actually, the corpse has been walking and talking for so long that it's a wonder no one has called Buffy Summers [1] to put a stake through its heart. However, one of our competitors did summon the spirit of *VirusTotal* (*VT*) to prove that AV is so far past its best-before date that it should only be used when given away free [2] (or maybe retrieved from the dustbins at the back of the cybermarket) .

The quasi-test – implemented by ignoring *VT*'s own recommendations and commentary on the misuse and misrepresentation of the service as a substitute for comparative testing [3] – actually tells us very little about real detection rates for the samples that were used, even assuming that they were valid samples of unequivocally malicious software. As *VT* rightly states: 'Those who use *VirusTotal* to perform anti-virus comparative analyses should know that they are making many implicit errors in their methodology.' I'll explain what some of those errors are later, but let's assume for the moment that *VT* is just being modest, and that a *VT* report is an accurate reflection of a product's detection performance (it isn't, and isn't meant to be).

At a time when AV labs process hundreds of thousands of samples a day, to claim on the evidence of 82 unverified samples that 'Anti-virus software is now so ineffective at detecting new malware threats most enterprises are probably wasting their money buying it' has more to do with marketing than with statistics. Since, by definition, we can't say what figure '100%' of known and unknown malware represents at any moment in time, we can't say what percentage of that totality is *detected* at any moment in time by any single AV product, let alone *all* products. We do know, though, that a very significant proportion of new threats are detected as soon as they appear by some form of code analysis and/or behaviour analysis. Of course, it's nowhere near 100%, or even the 80% that some AV vendors claimed for heuristics in the 1990s, and no AV researcher worth listening to would claim that it *is*, but it's a lot more than 0%.

If there's *any* single security solution (not just AV) that offers 100% detection and/or blocking of all malware and is easy and convenient to use, totally transparent to all business processes, and never generates any form of false positive, I wish someone would tell me what it is so I can go and buy a copy.

If this were a real test, I'd be sceptical of its accuracy because I don't know how the results were validated. We have no idea what samples were used (apparently acquired via TOR) or whether they were correctly classified as malware, still less about their prevalence. In the absence of that information, and of real testing that checks detection of validated samples against the whole functionality of the product (or at least both on-demand and on-access scanning) and using like-for-like configuration, there is more than a whiff of marketing about this exercise. Quasi-testing with *VirusTotal* is never going to accord with AMTSO's basic principles of testing [4] unless *VT* drastically re-engineers its mechanisms and objectives.

The fact is, *VT* was never intended as a mechanism for testing AV, and that is made very clear. A *VirusTotal* report doesn't tell you which solutions *know* about a specific threat sample. It tells you which (if any) solutions will flag it as a threat under very restricted conditions that don't reflect real-world conditions. If *VirusTotal was* meant as a tool (or a substitute) for comparative testing, it would be a very bad one.

But that isn't its purpose at all: it's meant to provide some idea as to whether a submitted file is malicious. (Even then the answer is equivocal: if enough vendors tell you it's malicious, the chances are it is, but if no vendor flags it as malicious, that doesn't mean it *isn't* malware.) *VirusTotal* is what it is – not a parable – but if you insist on describing it with an analogy, it's more like a heuristic scanner than a comparative test. A scanner with a tiny heuristic rule-set:

(Rule 1)

```
IF
One or more scanners flag file X as malicious or
suspicious
THEN
File X is definitely suspicious (but not proven malicious)
```

(Rule 2)

```
IF
No scanners flag file X as malicious or suspicious
THEN
File X is not suspicious (but could be malicious and
undetected)
```

OK, I'm being a little disingenuous here: *VirusTotal* does a lot more than that (and its full range of services is highly appreciated by the AV industry), but that's the functionality that is being cited by quasi-testers. (*VT*'s Julio Canto and I put together a paper [5] and presentation a couple of years ago for my favourite forensics conference [6] that covers what *VT* does in some detail, but also specifically addresses the issue of quasi-testing.)

VirusTotal is unsuitable for comparing product detection performance because the products it uses cover a wide range of functionality, and it doesn't configure all products to the same level of paranoia, or exercise all the layers of functionality they may comprise.

This means, for instance, that some products will flag potentially unwanted applications (PUAs) as malware: on some products, this is because of default settings, and in other cases, because *VT* has been asked by the vendor to turn on a non-default option. In other words, some products as configured by *VT* may never detect certain samples because they're not unequivocally malicious. If *VT* were a test, it would be more a test of vendor philosophy in terms of configuration parameters than a test of objective detection capability.

Other products may be able to detect a given sample on access, but not on demand, because not all approaches to behaviour analysis and heuristics can be implemented in static/passive scanning. *VirusTotal* uses command-line versions, and those versions do not implement whole product functionality because of the limited execution context of an essentially non-interactive scanner. To summarize the conclusion from that CFET paper [5]:

> *VirusTotal is a highly collaborative enterprise, allowing the industry and users to help each other. As with any other tool (especially other public multi-scanner sites), it's better suited to some contexts than others. It can be used for useful research or can be misused for purposes for which it was never intended, and the reader must have a minimum of knowledge and understanding to interpret the results correctly.*

Heuristics, generic detection, cloud technology: the AV industry has continuously attempted to adapt to changes in malicious technology and accelerating volumes of malware. What it hasn't done is communicate the extent to which anti-virus has ceased to be the product that it was decades ago – largely focused on detecting known virus samples (though even then, many products also had bundled integrity checkers, basically a form of whitelisting) – and has become a multi-layered product in its own right, incorporating several layers of defence. But anti-virus is not enough by itself, which is why mainstream products now incorporate their AV functionality into security suites. Sadly, they're not enough either, but then I'm not holding my breath waiting for a one-size-fits-all, never-needs-updating, 100% effective, never-gets-in-the-way-of-a-legitimate-process solution.

In principle, reputable security mavens advocate a sound combination of defensive layers rather than the substitution of one non-panacea for another. Actually, a modern anti-virus solution is already a compromise between malware-specific and generic detection, but I still wouldn't advocate anti-virus as a sole solution, any more than I would IPS, or whitelisting, or a firewall.

While some competitors in other industry sectors stop short of saying that people shouldn't use AV, they often suggest that there is no need to pay for it.

Vendors and journalists are actually doing their users/readers a disservice by suggesting that companies should use free AV so as to be able to afford another panacea du jour – not only because:

- that advice ignores the licensing stipulations that usually govern the legitimate use of free versions of commercial products;
- free AV has restricted functionality and support, especially when it's primarily a loss leader – a trailer to the main (for-fee) event;
- free AV has to make some other return on investment, which may take the form of strings attached in the form of complementary utilities, even adware [7].

But also because even where a security suite is the only security software in use, it offers more comprehensive, multi-layered protection than a product (free or otherwise) that only offers one layer of protection.

But can we imagine a world without AV, since apparently the last rites are being read already? A world in which vulnerability researchers get paid, but AV researchers don't, isn't altogether a pleasant prospect. While many of us do or have done a certain amount of *pro bono* work (in education and awareness raising, in standards organizations, and so on), most of us have to work for a living. It's unlikely that free AV would survive except among enthusiastic amateurs and companies in other security sectors throwing it in as a value-add, like those *Mac* utility vendors in the 90s who included detection of the few *Mac* viruses that existed at that time. Would the same companies currently dissing AV while piggybacking its research be able to match the expertise of the people currently working in anti-malware labs?

REFERENCES

[1] http://en.wikipedia.org/wiki/Buffy_the_Vampire_Slayer_%28TV_series%29.

[2] http://www.imperva.com/download.asp?id=324.

[3] https://www.virustotal.com/about.

[4] http://www.amtso.org/amtso---download---amtso-fundamental-principles-of-testing.html.

[5] http://go.eset.com/us/resources/white-papers/cfet2011_multiscanning_paper.pdf.

[6] http://www.canterbury.ac.uk/social-applied-sciences/computing/conferences/CFET2011/Home.aspx.

[7] http://blog.eset.com/2012/12/04/why-anti-virus-is-not-a-waste-of-money.

COMMENTARY 2

BREAKING DOWN BARRIERS FOR CYBERSECURITY: WHERE'S THE FIRST-MOVER ADVANTAGE?

Wout de Natris[1]
De Natris Consult, The Netherlands

Several news items that appeared in the first two weeks of December 2012 prompted me to start thinking in more depth about cybersecurity and international cooperation. This topic usually involves difficult discussions about cross-border jurisdiction issues, the need for cooperation between very different actors, and privacy. The term 'public-private cooperation' is often used, swiftly followed by 'we have to break down barriers'. If barriers need to be broken down, we should be asking: by whom, and what for? It may not be as easy to move forward as it seems. There may be conflicting interests among key players. Organizations may not be (fully) equipped to work outside of their primary remit. These issues are well known and I will not go into them, but what I do want to ask is: are there first-mover advantages when it comes to cooperation in cybersecurity? Can a collective action make a difference, and if so what could the first actions be? I will also discuss the concept of the Internet as a modern version of 'the commons'.

FOOD FOR THOUGHT

Over the course of a single day in December I read three news articles on the subject of cybersecurity. The first reported the news that Internet-connected smart TVs had been hacked for the first time (the first TV botnet is predicted to appear in 2013). The second story was about false QR codes in the public domain that led to malicious websites. The third was about Bluetooth-enabled skimming devices that were being used at gas stations in the US [1, 2]. In a matter of days, this was followed by news of built-in software that allowed for the skimming of card payment devices in stores, and news of a new *Android* botnet using the lure of free games to spread.

In a matter of days, there had been five new developments demonstrating the resourcefulness of criminals in taking advantage of flaws in network and standard security. This was on top of the near daily news about incidents that suggest that lessons are not being learned by those responsible for the security of networks, SCADA systems, new products that go online, etc.

[1] Inspiration on collective action theory provided by Joes de Natris.

THE INTERNET OF THINGS

When the Internet of things really starts happening, all sorts of devices will be connected to the Internet: refrigerators, coffee machines, car ports and who knows what else – maybe even the collars of our pet cats and dogs. Game consoles, TVs, air conditioners, printers, etc. are already connected to the Internet. Just imagine the possibilities for doing harm, from simple fraud to more malicious attacks. Are the industries that manufacture these appliances prepared for the security issues surrounding Internet connectivity?

The evidence so far seems to suggest not. The list of products that have been introduced to the market without basic protection against malicious activities is long. What is more concerning is that there does not seem to have been any progress in the form of lessons learned. The same mistakes are made over and over again, as demonstrated by the recent case of smart TVs being sold to consumers without basic protection. Meanwhile, in the mobile phone world, lessons learned the hard way by fixed line and ISP colleagues were not heeded at the switch from mobile operator to mobile ISP.

If more appliances are to come online in the (near) future, there is one link in the Internet chain that should be hoping for well secured appliances: the Internet Service Provider. Why?

CYBERSECURITY, CYBERCRIME AND ISPs

Recently, a representative of an Australian ISP walked out of official talks on anti-piracy, crying in outrage: 'We are not the Internet police!' [3]. This is just one example of the discussions ISPs get involved in. The ISP is seen more and more as a gatekeeper of the Internet: customers pay the ISP and receive Internet access in exchange. This puts the ISP in an awkward position as the only spot on the Internet where any end-user can easily be monitored and protected. Only, this is not the primary purpose of the ISP – like all businesses, its goal is to make a profit.

Governments increasingly turn to ISPs in their quest for greater security. For the ISPs, it is only the protection of their respective end-users (e.g. by filtering spam and malware or offering anti-virus products, etc.) that makes sense from the point of view of their business strategy. All other activities are costly, and as more and more duties are requested or made legal requirements by governments (e.g. botnet mitigation, monitoring, data retention, duty of care, reporting of incidents, etc.), ISPs may record a loss of profit. This puts them in a position in which they may want to influence certain discussions on Internet governance or reach out to certain parties in order to influence current developments. ISPs are not alone in this position of unused influence.

LARGE COMPANIES AND ORGANIZATIONS UNDER ONLINE ATTACK

We can see from the daily news that organizations are under constant digital attack: DDoS attacks, extortion, hacks, website infections, theft of vital business and customer data, industrial espionage, etc.

As far as I can ascertain, cybersecurity discussions among large organizations are mainly aimed at improving internal security. Organizations may work together as part of a financial or energy ISAC (Information Sharing and Analysis Centre), e.g. through a common early warning system and lessons-learned programme, but the focus for action is aimed at the individual companies. Of course such steps need to be taken, but are these the only ones possible for them to take in order to become part of a safer environment?

All companies, whether large or small, have a vested interest in a safer, more secure Internet. Yet, I do not see much evidence of participation by industry in vital talks surrounding the Internet and its security. What would the influence of large companies and institutions be, if they were to discuss and demand security on and around the Internet from their suppliers and others? What could the influence of ISPs be if they joined the discussion?

FIRST MOVER IS NOT AN ADVANTAGE

The Internet and the industry around it is extremely diverse. Making matters more complex is the fact that interests are also diverse, and even conflicting if we discuss Internet security in a general sense. (I'm not talking politics here, just economics.) For all commercial participants maximizing profits is key, but the maximization efforts of one party may infringe on the maximization of profits for another. For example, a company registering as many domain names as possible may infringe on the maximization of profit for an ISP, while, for example, partaking in measures to introduce a higher level of vetting prior to registration of a domain name does the same for the registrar (especially if he is the only one amongst his competitors to do so). To be a first mover is not an advantage in any of these cases, as it may impact business negatively. In some cases there may even be perverse incentives *not* to act, as money can be made as a by-product of cybercrime – for example, autodialling, SMS text fraud, direct messaging, registration of domain names for malicious intent and hosting the servers of spammers, all bring in revenues. In each case, a switch to an alternative provider can easily be made. A company that strives to be scrupulous not only has no first-mover advantage, but may lose business.

On the regulatory side, the situation is not much different. A study conducted by my consultancy [4] showed that it is hard for an organization to act beyond its primary task. Establishing national and international cooperation is not a primary task for enforcement agencies, CERTs, national centres on online threats, etc., and in times of budget restraint intensive cooperation is one of the first items to be dropped. It may also be abandoned if efforts at better cooperation are tangled in legal red tape, privacy issues, or if there is insufficient knowledge or will on the other side. Here too, first movers go unrewarded.

COLLECTIVE ACTION AND THE INTERNET

Collective action theory is about an individual making choices about whether to join a common cause. When does he participate, why, and are there rational reasons for not participating? When is it better to let someone else do something? Or no one? [5]. Go to almost any international meeting about Internet governance and you will hear 'We have to break down barriers!' But it is more interesting to ask: who is to break down barriers? Which barriers? What for? What should the outcome be? Where should we start? Do 'we' make anyone responsible for doing so?

Talking about an issue is the first step, but as I've tried to show above, there are conflicting interests between the potential participants, and no incentives or profits to reap for those that are willing to make the first move.

However, there are several examples in the Internet world where collective action has been taken. Organizations like ICANN, the Regional Internet Registries, IETF, IGF, M3AAWG, FIRST, etc., bring people together to discuss and work on specific topics. Some may participate because their bosses tell them to, others because they believe in the cause, others perhaps only because of the funded trips to appealing locations, but experience shows that there are people willing to put in extra work on committees that prepare discussions, guidelines, standards and (self-)regulations from which we all benefit.

There are also examples where organizations try to break down barriers through projects. For example, ENISA brings together police law enforcement agencies and CERTs on a regular basis to discuss (the challenges of) cooperation, with the specific aim of breaking down barriers. What role could the new European Cyber Crime Centre (EC3) play in the future? How is the Dutch National Cyber Security Centre doing having been in place for a year? What is the experience of MELANI in Switzerland, FICORA in Finland and Botfrei in Germany?

It's no longer just about work within one's own community. In the end, everyone profits from a safer Internet, as this leads to more trust and thus more development, innovations and business for all involved.

The analogy I would like to make is with 'the commons', the pieces of land for common use in the form of free grazing and foraging for wood, etc. in the Middle Ages or in the sea where fishing is concerned. No one owns the land, so no one feels responsible. The Internet seems like a modern form of the commons[2]. Although the consumer has to pay an entrance fee, after that he is in a limitless virtual environment, seemingly owned by no one, and no one on the consumers' side feels responsible. Just as over-grazing led to the end of the commons, abuse of the Internet will be the end of the Internet as we know it. Could a commonly felt responsibility for the Internet change the attitude of players on and around it?

SELF-REGULATION

I am aware that the word 'regulation' causes many Internet veterans to stand up and protest strongly. Still I want to debate it here. It is important to focus on what can be done to maintain the Internet as we know it, while at the same time making it safer and more robust.

If we look at the commons analogy, Shepsle and Bonchek [5] quote from Elinor Ostrom's book *Governing the Commons*. Overuse of the commons led either to a barren state that was of no use to anyone, or to a collective action in the form of self-regulation and severe self-restraint with (applied) sanctions and oversight. Times have changed since the commons of old, but Ostrom's studies may be an inspiration for discussions. In other, more modern examples, Shepsle and Bonchek look into how a government became involved and set up regulation. In the case of over-fishing this led to the involvement of the United Nations [5]. So what could the Internet world do, and how could governments be involved?

If regulating the Internet is not deemed acceptable, then self-regulation must be undertaken, even when there are conflicting interests. An example of a potential success story is the botnet mitigation centres or national centres on online threats. If all parties involved are willing to participate and act upon the warnings issued by these centres, self-regulation could become the accepted norm.

[2] This concept is not new, although I have not found references to the link with cybersecurity.

As long as the national (botnet) centre involved is neutral and issues warnings one-on-one, the centre is a trustworthy party for all concerned. It accepts, analyses and shares data in a neutral and non-discriminatory way among partners and non-partners alike. By taking action, participants close down the windows of opportunity that are available to attackers.

Within these centres, industry and law enforcement can cooperate as well. Data available through the centres may help to track cybercriminals and achieve convictions.

For those that fail to cooperate or fail to act upon warnings from the national centres, regulatory steps may be necessary. But how about assisting self-regulation without stifling progress?

A DUTY OF CARE

Even if the Internet industry, CERTs and enforcement agencies can break down the barriers between them through cooperation in national centres, this does not take care of the industries that manufacture products around the Internet. Software companies, appliance manufacturers, the gaming industry, banks, app stores, new digital payment systems, etc., must all be involved in the discussion about a safer Internet. Industry still delivers insecure products and companies buy insecure or insufficiently secured products, while at the same time being under attack from criminals, hackers, spammers, etc. – perhaps even through their own insecure products. This costs them large amounts of money. Could this not fuel the argument to get such companies involved in discussions surrounding a safer Internet? This will take time and patience, of course.

If governments do not wish to wait for this to happen, what could be an effective measure? Taking into account the fact that the Internet industry does not want regulation, it is necessary to look at more general measures. Specific regulation stifles all initiative, as one panellist at NLIGF's workshop at the last Internet Governance Forum stated: 'If you have a treaty or regulation that sets a bar, typically what businesses will do is to think "as long as I hit that regulation, I'm fine". Whereas, right now, you have people constantly striving to be better and have higher and higher bars.' [6].

At present, there is a continuous drive by companies like *Google*, *Microsoft*, anti-virus and commercial security vendors, etc. to come up with better security measures. However, this is not the case in a general sense. New products are insecure almost as a standard. I would like to see a general duty of care regulation imposed: a best practice regime in combination with the obligation to respond to incidents as well as notify a national agency (e.g. the Govcert) about them. In this way, the manufacturer learns from the incident and improves the practice, which becomes the new standard. Cybersecurity is a national priority these days, so why not impose a regulation that allows for best practices to be developed, employed and bettered within the Internet(-related) industry? Do you want to connect your product? Be sure to be secure! This way the prime initiative lies with industry, not the government, which can hold back, with a regulatory stick at hand when all else fails. The same should go for governmental and industry functions for the public good, like the security of SCADA systems, website, databases and privacy protection, etc.

If we add to this the notion that certain functions on and around the Internet are public, or at least very much in the public interest, but in private hands, the concept of responsible action, even in a highly competitive market, should not come as a surprise to those involved. Some measure of self-restraint may be called upon by society in a quest for a higher level of safety and security. And if this costs money, then we should all pay for the heightened security.

It seems that actions like these could be a middle ground for (preferably) self-regulation in favour of a more secure and free-flowing Internet that allows free speech, innovation and economic growth. A duty of care for Internet safety and security puts a non-discriminatory responsibility on all involved: those that provide access, host, distribute IP resources, manufacture software and hardware, deliver services on the Internet, connect to the Internet, etc. It could bring the Internet(-related) world, larger corporations and the public sector to the negotiating table, as they are invited to look at cybersecurity as a common challenge.

It will be of interest to see whether there is a vanguard of interested parties that are willing to lead in these discussions. People who are leaders in their respective communities and can take issues, discussions and actions back to their communities to discuss them further and assist in getting them implemented. Our world is changing quickly, and there are some in the Internet security community who are afraid that they will never catch up if cooperation and data sharing do not take place soon, and quickly [6]. The Internet is a great gift to society, industry and consumers alike. It is time to find a way to protect it, without losing its finer qualities.

CONCLUSION

In this article I have touched upon issues that at present prevent the very different entities involved in establishing a safer Internet from 'breaking down barriers'. Undoubtedly there is enough here to fill an academic study or two. By looking at the Internet as the modern commons, we can see that collective action is vital to protect it from abuse and crime. Industry needs to step beyond the inner security debate and start to reach out to and influence other players. It can, for example, play a role by setting up rulings for itself and self-regulate by active and responsive participation in national initiatives. For industry as a whole, there are enough incentives to participate in activities that make the Internet more secure, but it is necessary to approach the topic from more than one angle. The same goes for governments, institutions and (privatized) public functions.

Governments can play the role of last resort should self-regulation fail, and can coax industry, should it be necessary, to use more self-restraint or self-regulation. A general duty of care regulation imposed upon industry, government and public functions alike could do a lot to establish a level playing field which would create an environment in which there are first-mover advantages.

REFERENCES

[1] De Natris, W. One day, three new threats noted. International cooperation on cyber crime. The bridgebuilder's blog. http://woutdenatris.wordpress.com/2012/12/12/one-day-three-new-threats-noted/.

[2] De Natris, W. Cyber security. A duty to care? International cooperation on cyber crime. The bridgebuilder's blog. http://woutdenatris.wordpress.com/2012/12/13/cyber-security-a-duty-to-care/.

[3] ISP Walks Out of Piracy Talks: "We're Not The Internet Police". Torrent Freak. http://torrentfreak.com/isp-walks-out-of-piracy-talks-were-not-the-internet-police-121217/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+Torrentfreak+%28Torrentfreak%29.

[4] De Natris Consult report on (inter)national cooperation. http://woutdenatris.wordpress.com/2012/09/17/581/.

[5] Shepsle, K.A.; Bonchek, M.S. Analyzing Politics: Rationality, Behavior, and Institutions. W.W. Norton (1997).

[6] Internet Governance Forum (IGF). IGF Baku transcripts (Workshop 87). http://www.intgovforum.org/cms/2012-igfbaku/transcripts.

END NOTES & NEWS

Suits and Spooks DC takes place 8–9 February 2013 in Washington, DC, USA. For details see http://www.taiaglobal.com/suits-and-spooks/suits-and-spooks-dc-2013/.

RSA Conference 2013 will be held 25 February to 1 March 2013 in San Francisco, CA, USA. Registration is now open. For details see http://www.rsaconference.com/events/2013/usa/.

Cyber Defence Summit Middle East and N. Africa takes place 4–5 March 2013 in Muscat, Oman. For details see http://www.cyberdefencesummit.com/.

The 3rd Annual European Smart Grid Cyber and SCADA Security Conference takes place 11–12 March 2013 in London, UK. For more information see http://www.smi-online.co.uk/utility/uk/european-smart-grid-cyber-security.

Cyber Intelligence Asia 2013 takes place 12–15 March 2013 in Kuala Lumpur, Malaysia. For more information see http://www.intelligence-sec.com/events/cyber-intelligence-asia.

Black Hat Europe takes place 12–15 March 2013 in Amsterdam, The Netherlands. For details see http://www.blackhat.com/.

The 11th Iberoamerican Seminar on Security in Information Technology will be held 22–28 March 2013 in Havana, Cuba. For details see http://www.informaticahabana.com/.

EBCG's 3rd Annual Cyber Security Summit will take place 11–12 April 2013 in Prague, Czech Republic. To request a copy of the agenda see http://www.ebcg.biz/ebcg-business-events/15/international-cyber-security-master-class/.

SOURCE Boston takes place 16–18 April 2013 in Boston, MA, USA. For details see http://www.sourceconference.com/boston/.

The Commonwealth Cybersecurity Forum will be held 22–26 April 2013 in Yaoundé, Cameroon. For details see http://www.cto.int/events/upcoming-events/commonwealth-cybersecurity-forum/.

Infosecurity Europe will be held 23–25 April 2013 in London, UK. For details see http://www.infosec.co.uk/.

The 7th International CARO Workshop will be held 16–17 May 2013 in Bratislava, Slovakia. See http://2013.caro.org/.

The 22nd Annual EICAR Conference will be held 10–11 June 2013 in Cologne, Germany. For details see http://www.eicar.org/.

NISC13 will be held 12–14 June 2013. For more information see http://www.nisc.org.uk/.

The 25th annual FIRST Conference takes place 16–21 June 2013 in Bangkok, Thailand. For details see http://conference.first.org/.

CorrelateIT Workshop 2013 will be held 24–25 June 2013 in Munich, Germany. For details see http://www.correlate-it.com/.

Black Hat USA will take place 27 July to 1 August 2013 in Las Vegas, NV, USA. For more information see http://www.blackhat.com/.

The 22nd USENIX Security Symposium will be held 14–16 August 2013 in Washington, DC, USA. For more information see http://usenix.org/events/.

VB2013 will take place 2–4 October 2013 in Berlin, Germany. *VB* is currently seeking submissions from those wishing to present at the conference (deadline 8 March). Full details of the call for papers are available at http://www.virusbtn.com/conference/vb2013. For details of sponsorship opportunities and any other queries please contact conference@virusbtn.com.

SUBSCRIPTION RATES

Subscription price for Virus Bulletin magazine (including comparative reviews) for one 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.

Subscription price for Virus Bulletin comparative reviews only for one year (6 VBSpam and 6 VB100 reviews):

- Comparative subscription: $100

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/