SEPTEMBER 2011

virus BULLETIN

Fighting malware and spam

CONTENTS

ISSN 1749-7027

IN THIS ISSUE

CODE OF ETHICS

Having encountered a certain lack of cooperation among AV researchers on some major issues (perhaps due to issues of sharing data with competitors and non-disclosure agreements) Alex Eckelberry ponders whether it's time for an industry code of ethics.

page 2

PINS AND NEEDLES

While there is plenty of research on password use and re-use, there is virtually no equivalent research concerning purely numerical passcodes such as PINs. David Harley takes a look at some of the most common four-digit combinations used and the security issues raised.

page 12

OUT ON A LIMB

John Aycock takes the controversial view that Stuxnet is really not that interesting at all. He outlines what makes a piece of malware a game-changer and explains why last year's headline hitter is not worth writing home about.

page 14

'"Doing good for all is good for business" – helping others protect their users makes all of us stronger.'

Alex Eckelberry, GFI Software

SHOULD THERE BE AN AV INDUSTRY CODE OF ETHICS?

We see it all the time: a major magazine publishes a sensational story about a new nasty which threatens our existence; a researcher presents data at a major conference about a new threat; a company presents the inner workings of a group of black hats. The news machine grinds on it, while at the same time, a flurry of emails passes between researchers who are trying to gain more information on this 'threat'.

This is how the game is played. But recently, a number of security researchers have been questioning 'business as usual', and considering the industry's responsibility to share threat information with others. In other words, if we see something bad happening, what is our responsibility to do something about it?

It's worth noting that it is not uncommon for members of various security communities to share data, so at least other companies can protect their own customers, as well as collaborate on garnering further intelligence and even coordinate takedown efforts.

However, it appears that we may still be dealing with a lack of cooperation by a few on some major issues – perhaps due to issues of sharing data with competitors, non-disclosure agreements, or even defeatist ideas such as 'it doesn't really matter what we do anyway'.

Perhaps it's time that the security industry as a whole – not just the AV community – had a frank and open discussion about what our responsibilities are in protecting the community at large, in addition to promoting our own commercial interests. It's not an argument for 'malware welfare' – big, well capitalized companies sharing data with lesser capitalized companies. The fundamental issue is one that revolves around the need to make the Internet safer, rather than just pulling chips off the table.

The issue transcends a moral one that 'we have some duty to give back to the market if we're making money from it'. That's certainly a laudable imperative for many of us in the industry. However, I would argue toward a concept of enlightened self-interest, which could crudely be distilled as 'doing good for all is good for business' – helping others protect their users makes all of us stronger.

For example, spreading fear, uncertainty and doubt (FUD) among users, and then not doing what we can to make the Internet safer creates unintended consequences – we strike terror into the minds of users, and they demand solutions. This inevitably leads to politicians frantically trying to 'solve the problem'. More political action to 'regulate the dangers' is certainly not something about which many of us are sanguine.

My concern is that if we don't do what we reasonably can to keep the Internet clean, we will have regulatory agencies deciding to do it for us. Furthermore, users already distrust the security industry, and not working together to make the Internet safer will only lead to more scepticism. Finally, we know that having even a relatively small number of end-users that are unprotected against a threat can cause plenty of trouble for the rest of us. 'Every man for himself' is a losing strategy in the long term.

If we see something really bad, I would venture that it is in our commercial best interests to work with others to ensure their customers are protected, and to work as a community for intelligence sharing and takedown. If one company finds something bad, and wants a 'scoopable' news story, they can have it. Just share the data with others, so we can all make sure our customers are protected. We don't have to get frantic about every possible threat, but we can certainly focus on the major ones.

Is an industry code of ethics warranted? Perhaps, but in my view, we could simply start with promulgating industry best practices (codes of ethics, unless tied to some type of certification, are voluntary in nature anyway). I have found most security researchers to be honest, diligent folks who genuinely care about making the world safer. However, some may not be able to share data with competitors due to corporate policies, and they should not be in that position. Let's start with an honest, frank discussion about what what's good for all, and then perhaps what's good for us will come naturally.

NEWS

FACEBOOK BOUNTY HUNTERS PAID

Facebook has paid out a total of $40,000 in the first three weeks of its 'bug bounty program', in which researchers are given a financial reward for reporting new bugs found in the company's software code. The scheme was announced at the start of August, with a minimum payment of $500 for the disclosure of previously undiscovered security bugs.

In the first three weeks of the programme one individual was awarded more than $7,000 for having given the company a heads up on six different issues, while another received $5,000 for 'a really good report'.

In offering a reward for bug disclosure *Facebook* follows in the footsteps of other companies including *Google* and *Mozilla.* Last year *Mozilla* – whose bounty programme has been running for six years – increased the reward it pays for reports of security flaws in its software to $3,000 (plus a *Mozilla* T-shirt). Meanwhile *Google*, whose bounty programme was announced 20 months ago, pays a maximum of $3,133.7 for a single bug (but no T-shirt), with the base reward for less serious bugs remaining at $500.

Facebook has attracted criticism from within the security industry for its lack of monitoring of the third-party applications and websites that are built via the *Facebook Platform*, and has been asked whether it plans to extend the bounty programme to cover these. However, the company says that it would not be practical to do so because of the sheer number of third-party services implicated.

DROP IN CYBER ATTACKS

Symantec's 2011 State of Security survey has revealed a small drop in the number of businesses reporting cyber attacks. In this year's survey 71% of respondents said they had experienced attacks in the past 12 months, compared with 75% in 2010. Meanwhile, the number of respondents reporting an increase in frequency of attacks dropped from 29% to 21%.

Despite this, cyber attacks remain the top concern for the businesses surveyed for the second year running – ahead of traditional crime, natural disasters and terrorism – and 41% of respondents said they felt that cybersecurity is more important now than it was a year ago.

The survey also revealed a drop in the losses experienced as a result of cybercrime, with 92% of organizations reporting losses from attacks, compared with 100% last year. The top three losses reported were downtime, theft of employees' personal information and theft of intellectual property. Among SMBs, 20% incurred financial losses of at least $100,000 from attacks within the last year, while 20% of larger enterprises incurred at least $271,000 in damages.

Prevalence Table – July 2011 [1]

Malware	Type	%
Autorun	Worm	9.60%
FakeAlert/Renos	Rogue	6.65%
VB	Worm	5.68%
Heuristic/generic	Misc	5.50%
Adware-misc	Adware	5.10%
Conficker/Downadup	Worm	4.13%
OnlineGames	Trojan	4.07%
Sality	Virus	4.03%
StartPage	Trojan	3.62%
Agent	Trojan	3.58%
Iframe	Exploit	2.84%
Downloader-misc	Trojan	2.68%
LNK	Exploit	2.30%
Injector	Trojan	2.22%
Delf	Trojan	2.14%
AutoIt	Trojan	1.82%
Heuristic/generic	Trojan	1.81%
Zbot	Trojan	1.71%
Virtumonde/Vundo	Trojan	1.69%
Crack/Keygen	PU	1.51%
Dropper-misc	Trojan	1.43%
Virut	Virus	1.37%
Alureon	Trojan	1.24%
Kryptik	Trojan	1.19%
Hotbar	Adware	1.06%
Crypt	Trojan	1.05%
Bifrose/Pakes	Trojan	1.01%
Small	Trojan	0.90%
MyWebSearch	Adware	0.87%
Themida	Packer	0.87%
BHO/Toolbar-misc	Adware	0.85%
Dorkbot	Worm	0.83%
Others [2]		14.62%
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

'HOLEY' VIRUS, BATMAN!

Peter Ferrie
Microsoft, USA

Some might think that all of the entrypoints in Portable Executable (PE) files are known – but they would be wrong. As we saw with the W32/Deelae family [1], a table that has been overlooked for more than a decade can be redirected to run code in an unexpected manner. Now, a table that was used in *Windows* on the *Itanium* platform also exists on the x64 platform, and (surprise!) it can be misused too. The W64/Holey virus shows us how.

HAPI HAPI, JOY JOY

The virus begins by retrieving the address of ntdll.dll by walking the InMemoryOrderModuleList list from the PEB_LDR_DATA structure in the Process Environment Block. This is an unusual choice – the InLoadOrderModuleList list is more common – but it is not incorrect, and it is compatible with the changes that were made in *Windows 7*. The virus also saves the pointer to the current position in the list so that it can resume the parsing later to find the address of kernel32.dll.

If the virus finds the PE header for ntdll.dll, it resolves the required APIs. It uses hashes instead of names, but the hashes are sorted alphabetically according to the strings they represent. This means that the export table needs to be parsed only once for all of the APIs, rather than parsed once for each API (as is common in some other viruses). Each API address is placed on the stack for easy access, but because stacks move downwards in memory, the API addresses end up in reverse order in memory. Interestingly, the virus checks that the exports really exist by limiting the parsing to the number of exports in the table. This is probably a great situation for emulators that don't export all of the right functions – the sample will run the host code instead of crashing – but it doesn't benefit the virus in any way.

The table is terminated with a single byte whose value is 0x2a (the '*' character). This is used to allow the file mask to follow immediately in the form of '*.exe'. We do not know whether the character was chosen because of the mask, or whether the mask was placed there simply because of the chosen character.

The virus retrieves the address of kernel32.dll by fetching the next entry in the list, using the pointer that was saved earlier. The same routine is used to retrieve the addresses of the API functions that it requires. Despite the strong similarities with some other viruses that support Unicode, this virus only uses ANSI APIs. The result is that some files cannot be opened because of the characters in their names, and thus cannot be infected.

The virus searches in the current directory (only), for files whose names end in '.exe'. For each such file that is found, the virus attempts to open it and map a view of the contents. There is no attempt to remove the read-only attribute, so files that have that attribute set cannot be infected. The virus registers an exception handler at this point, and then checks if the file can be infected.

RELOCATION ALLOWANCE

The virus is interested in Portable Executable files for the x64 platform. Renamed DLL files are not excluded, nor are files that are digitally signed. The subsystem value is checked, but incorrectly. The check is supposed to limit the types to GUI or CUI but only the low byte is checked. Thus, if a file uses a (currently non-existent) subsystem with a value in the high byte, then it could potentially be infected too. In fact, there are many common checks that are missing from this virus. Perhaps it was written in a hurry to meet some kind of deadline. The code also lacks some obvious optimizations, again suggesting that it was written hastily.

The virus checks the Base Relocation Table data directory to see if the relocation table is the last section. The check is very specific – it is not enough that the relocation table exists *within* the last section, but it must *be* the last section. That is, it starts at the start of the section, and the assumption is that the entire section is devoted to relocation information – which can cause a problem. The virus checks that the value in the SizeOfRawData field is at least 687 bytes long. Of course, the relocation table could be much smaller than this, and other data might follow it. This data will be overwritten when the virus infects the file.

We do not know why the SizeOfRawData field was used instead of the value in the Size field in the data directory, because the value in the Size field cannot be less than the size of the relocation information, and it cannot be larger than the size of the section. This is because the value in the Size field is used as the input to a loop that applies the relocation information. It must be at least as large as the sum of the sizes of the relocation data structures. However, if the value were larger than the size of the relocation information, then the loop would access data after the relocation table, and that data would be interpreted as relocation data. If the relocation type was not a valid value, then the file would not load. If the value in the Size field were less than the size of the relocation information, then

it would eventually become negative and the loop would parse data until it hit the end of the image and caused an exception.

EXCEPTIONAL BEHAVIOUR

The virus also requires that the file has no Exception Table. If this is the case, then the virus creates a RUNTIME_FUNCTION structure and places it at the start of the last section. The RUNTIME_FUNCTION structure contains the begin and end addresses of the code which will be described by the UNWIND_INFO structure, and a pointer to that structure. The virus sets the begin address to equal the host entrypoint value, and the end address to one byte later than that. The UNWIND_INFO structure pointer is set to the address immediately after the pointer, and the UNWIND_INFO structure is placed directly after the RUNTIME_FUNCTION structure. The UNWIND_INFO structure exists to allow *Windows* to unwind the stack if an exception occurs. However, the virus does not need to worry about such a thing. All it has to do is set the appropriate flags and store a callback pointer in the structure. Then, when an exception occurs, the virus code will be called.

The virus makes the last section both writable and executable. It sets the Exception Table data directory entry to point to the start of the last section, and sets the Size field appropriately. The virus copies itself to the file, and zeroes some flags in the header. Of particular interest are the flags that correspond to ASLR (Address Space Layout Randomization), NX (No eXecute) and NO_SEH (No Structured Exception Handling). Zeroing the ASLR flag ensures that the image will not move in memory. This is irrelevant for the virus, though, because the virus code is entirely position-independent. It might be a bit of left-over code from a previous virus by the same author. Zeroing the NX flag enables the virus to run from a section that is not marked as executable. Again, this is irrelevant for the virus because the virus marks its code section as executable. However, by zeroing the NO_SEH flag, the virus enables exception handling to be called by the file. If the flag were not cleared, then *Windows* would terminate the application at the moment that an exception occurred.

The virus zeroes the Base Relocation Table data directory. This is the only way to ensure that *Windows* does not attempt to read the relocation data. Even though there exists a flag that can be set in the PE header which is supposed to tell *Windows* that the relocation data has been removed, *Windows* ignores this flag in certain circumstances.

YOU HAVE BEEN INTERRUPTED

At this point the virus attempts to find the section that contains the entrypoint by searching for the first section which ends after the entrypoint. There are two bugs here, one is relatively minor, however the other is fatal for the host. The minor bug is that the virus assumes that the entrypoint is located within a section. It is quite possible to place the entrypoint in the file header. It is possible to place the entrypoint outside of the image, and through a bit of trickery, cause it to 'move' back inside the image (the details about how this is done are not relevant here). It is possible for the file to contain no sections, but still have the appropriate values in the appropriate places. Of course, these are edge cases that are not very interesting to consider.

However, the fatal bug relates to the section scanning. When the virus finds a section which ends after the entrypoint, it marks that section as writable and attempts to fetch the byte at the relative virtual address that corresponds to the value of the entrypoint. The byte is saved in the virus body, and replaced with an interrupt 3 instruction. The idea here is that when the host is executed, the interrupt 3 instruction will cause an exception that will be handled by the callback whose pointer is in the Exception Table. The bug is that even after the first such section is found (the entrypoint section), the loop is not exited. Instead, every section after the entrypoint section will also be treated as though it were the entrypoint section. What happens next depends on several conditions. The entrypoint value is converted to a physical address by adding the difference between the PointerToRawData and the VirtualAddress. The addition will occur for each section after the entrypoint section, usually resulting in a continually decreasing value that still points within the entrypoint section. If the value becomes negative (for example, if the initial entrypoint value is small, and the difference is large), then an exception will occur when attempting to fetch the byte from the section. The exception will cause the loop to exit, and everything will appear to be fine – this is probably what happened when the virus was tested, and is probably the reason the bug was not found. However, if the entrypoint value is large enough and if the difference is small enough (it can even be zero, if the file alignment value matches the section alignment value), then the value can survive several, and possibly all of the iterations of the loop, resulting in multiple bytes being replaced in the host entrypoint section.

The bug leads to two other problems, one of which is benign, and the other one causes the host to be damaged. The first problem is that when the last section is examined,

if the entrypoint RVA is larger than the size of that section, then an exception will occur and the infection routine will exit. This is fine because when the loop completes, an exception is raised anyway. The second problem is that, as noted above, the virus might alter one byte in multiple places within the entrypoint section. If those bytes are not all the same (or if they are, but the file alignment and section alignment match, such that the same byte is fetched more than once), then when the exception occurs during execution, the original byte cannot be restored. Further, if those bytes are data, then the host might not behave correctly even if the bytes are all the same because the virus will never have a chance to restore them.

TOUCH AND GO

The virus code ends with an instruction to force an exception to occur. This is used as a common exit condition. However, the virus does not recalculate the file checksum, even though it might have changed as a result of infection, and it does not restore the file's date and timestamps, making it very easy to see which files have been infected, even though the file size does not change.

ANCIENT HISTORY

It's funny, in a way, that I described a variation of this technique at a time when *Windows NT* was still current. The idea was that by changing the file format in a particular way, an exception would be raised during the file *load*. At that point, nothing in the host had been executed. Given an Exception Table with the right layout, it should have been possible to cause the handler to be executed. (Try emulating that...) However, as noted above, the Exception Table was not used by *Windows* until the introduction of the *Itanium* platform, so fortunately the technique was not viable.

CONCLUSION

The Exception Table hook is an interesting technique. It allows for light entrypoint obscuring, in much the same way as the Thread Local Storage technique did a decade ago, and it becomes yet another place in the file that needs to be scanned. Only time will tell if it will become as popular.

REFERENCES

[1] Ferrie, P. Deelaed learning. Virus Bulletin, November 2010, p.8. http://www.virusbtn.com/pdf/magazine/2010/201011.pdf.

FEATURE 1

QAKBOT: A DISASTER WAITING TO HAPPEN

Jessa Dela Torre
Trend Micro, Philippines

The recent security breaches at the Massachusetts Departments of Unemployment Assistance and Career Services [1] revived public interest in Qakbot (aka Qbot or Pinkslipbot), the malware family that has been responsible for infiltrating thousands of systems worldwide during its four-year stint [2].

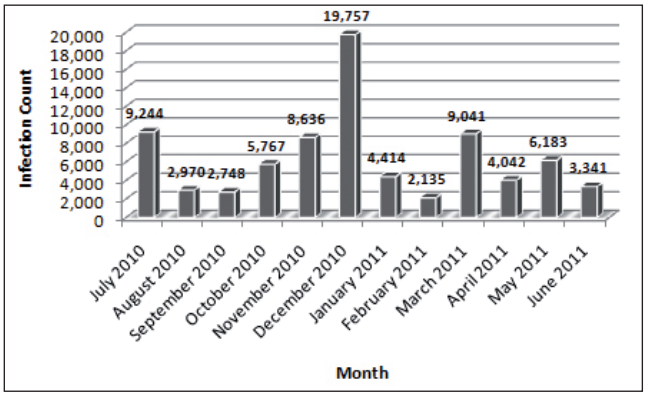


Figure 1: Qakbot infection count, July 2010 – June 2011 (as of 5 June 2011).

At *Trend Micro* we have received several escalations with regard to this particular malware family from various enterprise customers in the healthcare, financial, government, and other sectors.

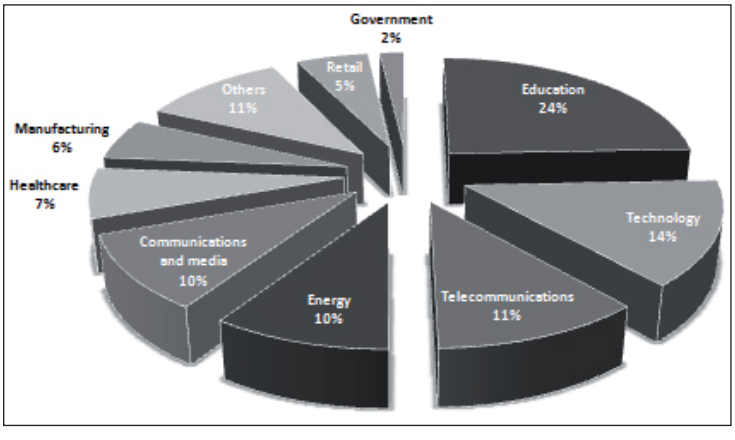


Figure 2: Qakbot infection count breakdown by industry, July 2010 – June 2011 (as of 21 June 2011).

We came across a new Qakbot variant in the latter part of February this year [3]. Like its predecessors, this variant

propagates primarily via network shares. The malware has also undergone a structural revamp and added a new infection vector, so it came as no surprise when news of massive Qakbot network infections broke. Armed with more effective means of infection, the malware quickly spread worldwide, leaving an indelible mark on the threat landscape.

Qakbot's ability to propagate via network shares is enough to cripple an entire network. Add to that the ability to compromise websites and you seemingly have a recipe for a highly successful malware attack.

This paper will discuss the different ways in which Qakbot variants arrive on and infect systems, how these affect users, and how the security industry can help mitigate Qakbot system infections.

QAKBOT MALWARE EVOLUTION

First-generation Qakbot variants can be distinguished based on their file and folder names, which usually include the string '_qbot'. These store their components in password-protected .ZIP files that their main components download from certain sites.

To make system infections less obvious, next-generation Qakbot variants started using random file and folder names. Their package files also ditched password protection and started taking the social engineering route, using file names such as 'resume.doc'.

As the security industry caught on and started being able to detect Qakbot successfully, the malware's creators introduced another new breed of variants with a few significant changes.

First, they no longer made use of an archive file, instead packaging every component in a single .EXE file. Next, they added a new propagation vector: USB drives.

KNOWN QAKBOT INFECTION VECTORS

Early Qakbot variants exploited software vulnerabilities in order to infect systems. They particularly took advantage of the Collab.collectEmailInfo and Collab.getIcon vulnerabilities in certain versions of *Adobe Acrobat* and *Adobe Reader* via malicious PDF files. More recent variants have been found to arrive via three infection vectors: removable drives, default network shares and as drive-by downloads from compromised sites.

Removable drives

The USB port is perhaps any system's weakest point in terms of network security since no firewall or network policies can be enforced to maintain the integrity of a removable drive. Typically, office employees use USB drives when performing their daily tasks. This is probably the reason why cybercriminals continue to use malware that spreads via removable drives. In fact, so-called USB malware such as the Palevo [4], Qakbot [5] and Vobfus [6] worms continues to reign on *Trend Micro*'s list of most notorious malware types.

Earlier Qakbot variants did not have the ability to spread via removable drives. The malware's authors probably realized how successful other USB malware had been and decided

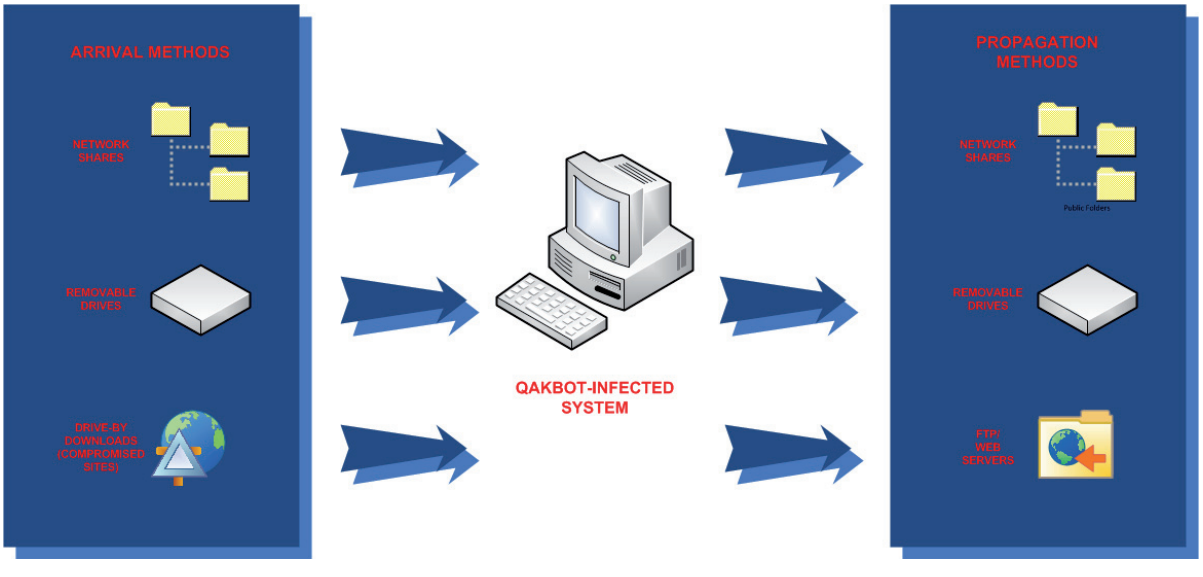


Figure 3: Typical Qakbot infection diagram.

to include the ability to spread via removable drives in more recent versions of Qakbot.

Unlike other USB malware, however, Qakbot variants do not use an autorun.inf file. Instead they rely on crafty social engineering techniques and on the tendency for users to unwittingly execute files.

Whenever a USB drive is plugged into an infected system, the Qakbot variant will select a file from the drive's contents and drop a copy of itself onto the system using the chosen file's name in the following format (Figure 4):

{malware's file name}_{selected file's name}.exe

If the USB drive is empty, the malware will just append '_Documents' to its own file name (Figure 5):

{malware's file name}_Documents.exe

Once clicked, the copy of the Qakbot variant will then perform its malicious routines. If this happens to a system that is connected to a network, the Qakbot infection will spread to all of the systems on the network.

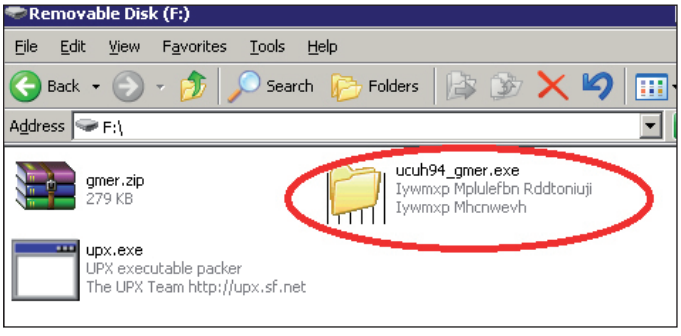


Figure 4: Malware file in an infected USB drive.

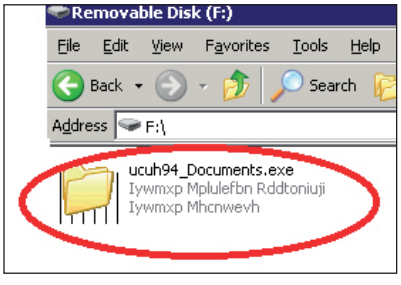


Figure 5: Copy of the malware that will be dropped from the infected USB drive onto the user's system.

Default network shares

The default administrative network shares on *Windows* are created by each system. As such, deleted shares will reappear whenever a system reboots. Qakbot variants make use of these default shares to propagate across a network. To do so, they initially enumerate all of the available resources. They then attempt to establish connections based on the affected user's rights.

Before dropping a copy of itself onto a system, the Qakbot variant will first check the following:

- Whether the resource belongs to its current host

1	0.000000	192.168.1.4	192.168.1.1	TCP	casp > microsoft-ds [SYN] Seq=0 Win=64240 Len=0 MSS=1460
2	0.000157	192.168.1.1	192.168.1.4	TCP	microsoft-ds > casp [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MS
3	0.000187	192.168.1.4	192.168.1.1	TCP	casp > microsoft-ds [ACK] Seq=1 Ack=1 Win=64240 Len=0
4	0.000491	192.168.1.4	192.168.1.1	SMB	Negotiate Protocol Request
5	0.001099	192.168.1.1	192.168.1.4	SMB	Negotiate Protocol Response
6	0.010005	192.168.1.4	192.168.1.1	TCP	[TCP segment of a reassembled PDU]
7	0.010078	192.168.1.4	192.168.1.1	TCP	[TCP segment of a reassembled PDU]
8	0.010140	192.168.1.4	192.168.1.1	SMB	Session Setup AndX Request
9	0.010177	192.168.1.1	192.168.1.4	TCP	microsoft-ds > casp [ACK] Seq=159 Ack=3058 Win=64240 Len=0
10	0.011678	192.168.1.1	192.168.1.4	SMB	Session Setup AndX Response
11	0.012120	192.168.1.4	192.168.1.1	SMB	Tree Connect AndX Request, Path: \\[redacted]\C$
12	0.013135	192.168.1.1	192.168.1.4	SMB	Tree Connect AndX Response
13	0.217584	192.168.1.4	192.168.1.1	TCP	casp > microsoft-ds [ACK] Seq=3770 Ack=652 Win=63589 Len=0
14	1.467465	192.168.1.4	192.168.1.1	SMB	NT Create AndX Request, Path: \ekxvrskxd.exe
15	1.468720	192.168.1.1	192.168.1.4	SMB	NT Create AndX Response, FID: 0x4000
16	1.468835	192.168.1.4	192.168.1.1	SMB	Trans2 Request, QUERY_FILE_INFO, FID: 0x4000, Query File Inte
17	1.469044	192.168.1.1	192.168.1.4	SMB	Trans2 Response, FID: 0x4000, QUERY_FILE_INFO
18	1.528527	192.168.1.4	192.168.1.1	SMB	Trans2 Request, QUERY_FS_INFO, Query FS Attribute Info
19	1.529045	192.168.1.1	192.168.1.4	SMB	Trans2 Response, QUERY_FS_INFO
20	1.532458	192.168.1.4	192.168.1.1	SMB	Trans2 Request, QUERY_FILE_INFO, FID: 0x4000, Query File Basi
21	1.532716	192.168.1.1	192.168.1.4	SMB	Trans2 Response, FID: 0x4000, QUERY_FILE_INFO
22	1.537392	192.168.1.4	192.168.1.1	SMB	Trans2 Request, SET_FILE_INFO, FID: 0x4000
23	1.538977	192.168.1.1	192.168.1.4	SMB	Trans2 Response, FID: 0x4000, SET_FILE_INFO
24	1.590770	192.168.1.4	192.168.1.1	TCP	[TCP segment of a reassembled PDU]
25	1.590856	192.168.1.4	192.168.1.1	TCP	[TCP segment of a reassembled PDU]
26	1.590920	192.168.1.4	192.168.1.1	TCP	[TCP segment of a reassembled PDU]
27	1.590975	192.168.1.1	192.168.1.4	TCP	microsoft-ds > casp [ACK] Seq=1111 Ack=7122 Win=64240 Len=0
28	1.591000	192.168.1.4	192.168.1.1	TCP	[TCP segment of a reassembled PDU]
29	4.982209	192.168.1.4	192.168.1.1	DCERPC	Bind: call_id: 1 SVCCTL V2.0
30	4.982763	192.168.1.1	192.168.1.4	SMB	Write AndX Response, FID: 0x4001, 72 bytes
31	4.982947	192.168.1.4	192.168.1.1	SMB	Read AndX Request, FID: 0x4001, 1024 bytes at offset 0
32	4.983192	192.168.1.1	192.168.1.4	DCERPC	Bind_ack: call_id: 1 accept max_xmit: 4280 max_recv: 4280
33	4.983382	192.168.1.4	192.168.1.1	SVCCTL	OpenSCManagerA request, \\[redacted]
34	4.983922	192.168.1.1	192.168.1.4	SVCCTL	OpenSCManagerA response
35	4.984967	192.168.1.4	192.168.1.1	SVCCTL	CreateServiceA request
36	5.136054	192.168.1.1	192.168.1.4	TCP	microsoft-ds > casp [ACK] Seq=2969 Ack=293052 Win=62834 Len=0
37	5.303877	192.168.1.1	192.168.1.4	SVCCTL	CreateServiceA response
38	5.313974	192.168.1.4	192.168.1.1	SVCCTL	StartServiceA request
39	5.443604	192.168.1.1	192.168.1.4	TCP	microsoft-ds > casp [ACK] Seq=3081 Ack=293192 Win=64240 Len=0
40	7.163123	192.168.1.1	192.168.1.4	SVCCTL	StartServiceA response
41	7.306271	192.168.1.4	192.168.1.1	TCP	casp > microsoft-ds [ACK] Seq=293192 Ack=3169 Win=64040 Len=0

Figure 6: Wireshark capture of the protocol used by Qakbot.

- Whether the target host is already infected (by checking the nbl section of its configuration file).

If both are untrue, the malware will drop a copy of itself onto C$ or Admin$ – both of which are default shares – and start a remote service in order to execute the file it dropped. It uses SMB over TCP to access the target resource and to send a copy of itself to connected systems.

The Qakbot variant then binds the SVCCTL interface to start a remote service using the following command:

%path%\{file name}.exe /s

Drive-by downloads via compromised sites

Since first-generation Qakbot variants reared their ugly heads, part of their distribution routine has been the ability to inject malicious scripts into files stored on web servers. To do this, they initially attempt to contact a command-and-control (C&C) server in order to get a command file that is saved as %User Temp%\~{random name}.tmp. This file contains the FTP credentials the malware needs in order to perform its script injection routine. Once connected, it downloads files with the following extension names:

- .php
- .htm
- .asp
- .pl
- .cfm

The malware then inserts a malicious script before the </body> tag, which serves as a link to one of its download sites. It then re-uploads the infected files to the web servers. These then effectively infect the systems of users who visit compromised sites. The download link can lead to a copy of the main Qakbot executable file or to its JavaScript component.

QAKBOT INFECTION IMPLICATIONS

Once inside a system, Qakbot monitors substrings related to financial institutions such as *Bank of America*, *Fifth Third*, *Wells Fargo* and *Citibank*, among others. The malware also gathers the following information:

- System information
- IP address

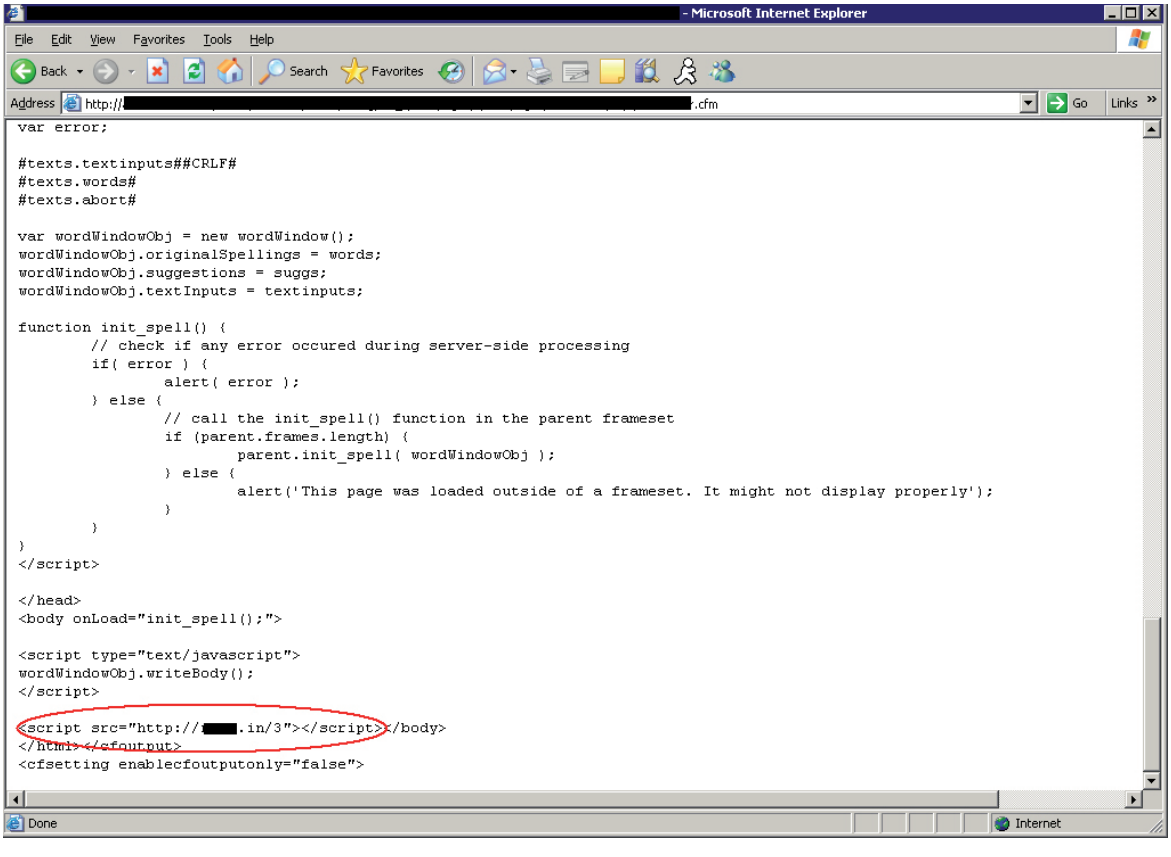


Figure 7: Code of a web page that has been injected with a malicious script.

- Domain Name System (DNS) name
- Host name
- User name
- Domain
- User privilege
- OS version
- Network interfaces (i.e. address, netmask and status)
- Installed software
- Protected storage
- Account name
- Connection type
- POP3 (i.e. username, server and password)
- SMTP (i.e. server and email addresses)
- *Internet Explorer* (*IE*) and *Flash Player* cookies
- Certificates
- Web server credentials (i.e. usernames and passwords)
- Keystrokes

All of the aforementioned information is sent to Qakbot-controlled FTP sites. Sometimes the data in these FTP sites remains for a few days and builds up to quite a sizeable amount.

Given the type of data Qakbot collects, an organization stands to lose a lot of business-critical information related both to the organization itself and its clients. Data exposure not only leads to bad publicity but can also plant a sense of distrust in an organization's ability to protect its resources.

The bulk of responsibility falls into the hands of system administrators. Securing a network and enforcing strong policies are just some of the things they have to worry about. Cleaning up after system or network infections is another story. A Qakbot-infested network will require a lot of work and specialized tools to remove the malware.

The extent of Qakbot infections does not rely on state-of-the-art propagation routines. Security researchers may even argue that Qakbot's method of spreading is not new and is certainly not that complex. The variants do not exploit zero-day or even run-of-the-mill vulnerabilities. Instead they bank on social engineering tactics and poor network security – both of which can be prevented if system administrators are more aware and better prepared for malicious attacks.

Qakbot variants were originally designed to infiltrate systems for the sole purpose of gathering as much information as possible. They currently utilize three of the most widely used infection vectors. Improved strains also ensure greater revenue for malware authors, which may eventually lead to a new batch of variants with even better functionality.

The most recently discovered Qakbot configuration files have an entry dubbed peer-to-peer (P2P) node list, which points to a URL. The URL is not currently accessible, but we can expect the link to be activated as soon as a new batch of Qakbot variants is rolled out.

SAFE COMPUTING PRACTICES TO AVOID QAKBOT INFECTIONS

The following safe computing practices can help system administrators protect their organizations' networks against Qakbot-related attacks.

Limit users' administrative rights

Qakbot's ability to spread via default network shares allows it to instigate mass network infections. System administrators should keep in mind that the malware's access to network resources is dependent on the affected users' system privileges. As such, they should limit users' administrative rights and should issue write permissions conservatively. Users should only be assigned the lowest level of privilege required to complete their tasks. Password protecting network shares is also a good idea to combat the threats Qakbot poses.

Educate users

It is important to educate users about securing and utilizing company resources properly, particularly removable drives and Internet access.

We have seen several instances in which Qakbot variants have infected systems via removable drives. Security awareness is a critical element of safe computing.

Encourage users to practise safe web browsing habits

Since Qakbot has the ability to easily compromise sites, the fact that a site is visited frequently or well known is no guarantee of its safety. In fact, popular sites may even present greater risk.

Users must be extra careful when clicking links. Some security solutions feature a web reputation service functionality which will block access to malicious sites and mitigate web-based attacks. Figure 8 shows *Trend Micro*'s web reputation service blocking the web browser from visiting a known malicious site.

Google's *Safe Browsing* service [7] is another useful tool for safer web browsing. This and similar services provide information on a site's integrity, including its involvement, if any, with malicious activities (Figure 9).

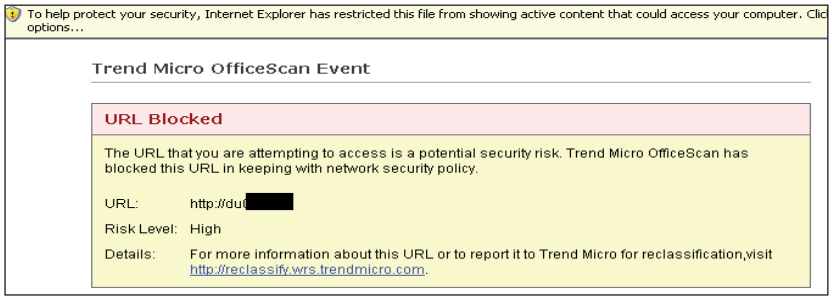


Figure 8: Trend Micro's web reputation service feature at work.

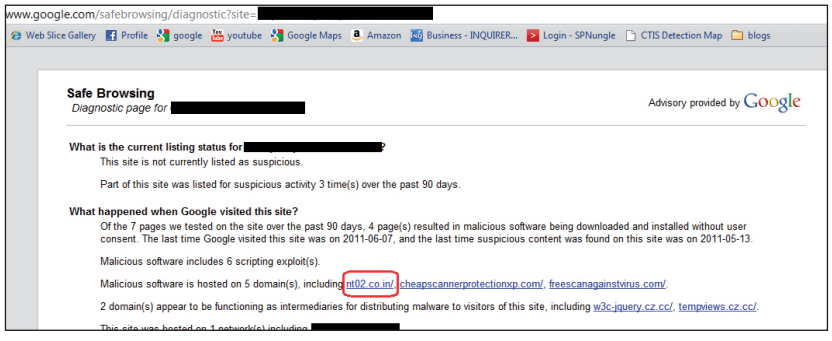


Figure 9: Qakbot-infected site assessed by Google's Safe Browsing service.

Update anti-virus signatures and install OS and software patches

As malware authors continually improve their malicious wares to evade even the best scanners, so must security firms constantly update their anti-virus signatures to mitigate Qakbot infections.

System administrators are advised to download and install security patches as soon as they are made available. This will help defend systems and networks against vulnerability exploit attacks.

Install a network monitoring device

System administrators are strongly advised to install security appliances that will help them monitor network activities.

Several kinds of intrusion detection systems (IDS) and intrusion prevention systems (IPS) are available on the market. System administrators should ensure that the devices they install on their company networks are capable of identifying, blocking and reporting any and all kinds of suspicious activities.

Even though these practices may not make a system or a network Qakbot-proof, they will definitely lower the chances of infection. Given that Qakbot is after almost every kind of data in a system or in a network, following the aforementioned tips certainly would not hurt.

CONCLUSION

Qakbot variants are primarily known for two things – their huge appetite for confidential data and their ability to rapidly spread across networks – the combination of which can spell disaster for affected users.

Qakbot infections not only put the affected user's personal information at risk, they also put corporate data on the affected user's system or on the network in grave danger of being compromised. The recent enhancements to Qakbot's distribution routines have immensely increased the malware's notoriety.

Network security plays a pivotal role in containing Qakbot infections. Good heuristic or generic signatures can prevent the malware from running on systems and from spreading across networks. Note, however, that since Qakbot variants are known for changing their structure, file-based detection alone may prove ineffective. Using a combination of file- and behaviour-based detection 'in the cloud' [8] may, in the end, be the best solution to counter the ever-persistent Qakbot threat.

REFERENCES

[1] http://www.theregister.co.uk/2011/05/20/massachusetts_worm_infection/.

[2] http://about-threats.trendmicro.com/RelatedThreats.aspx?language=us&name=Qakbot%3a+A+Prevalent+Infostealing+Malware.

[3] http://blog.trendmicro.com/third-generation-qakbot-repackaged-with-improved-propagation/.

[4] http://blog.trendmicro.com/mariposapalevo-on-the-rise-again/.

[5] http://about-threats.trendmicro.com/RelatedThreats.aspx?language=us&name=Qakbot%3a+A+Prevalent+Infostealing+Malware.

[6] http://about-threats.trendmicro.com/search.aspx?language=us&p=WORM_VOBFUS.

[7] http://code.google.com/apis/safebrowsing/.

[8] http://us.trendmicro.com/us/trendwatch/core-technologies/smart-protection-network/.

FEATURE 2

HEARING A PIN DROP

David Harley
Small Blue-Green World/Mac Virus, UK

One thing has become painfully obvious in the light of the recent spate of attacks and data leakages, not to mention various LulzSec nautical naughtiness. Clearly, people are continuing to use highly stereotyped password strategies: in other words, many, many people are using a very, very small selection of passwords. But while there's plenty of research on password use and re-use – mostly derived from the analysis of known collections of exposed passwords [1] to see which are the most commonly used – there is virtually no equivalent research concerning purely numerical passcodes such as PINs (Personal Identification Numbers). While there are no high-profile and publicly available repositories of leaked PIN data allowing empirical analysis (scraping underground forums presents both practical and ethical problems), there is one recent instance [2] of research based on analysis of smartphone passcodes, though it's not the result of a LulzSec-type breach.

Daniel Amitay has been marketing an app [3] called Big Brother (social networking meets reality TV?) – intended to take photos of anyone using an *iPhone* or *iPod Touch 4* without permission (i.e. without entering a passcode). A recent update to the app added code that captures (completely unidentifiably, he promises) the passcodes entered during set-up of the app. This enabled Amitay to run some analysis on a sample set of 204,508 gadgets. These particular iGadgets offer a choice of passcode modes for screenlocking: off, simple four-digit passcode, or a more complex passcode. While we cannot assume that a choice of passcode for Big Brother would reflect either screenlocking passcode selection or PIN selection practice, it seems reasonable to assume that, given the size of the sample, there is likely to be some correlation. (*Apple* clearly thought so, since it has removed the app from the *App Store* and insisted that the passcode-recording code be removed before it is reinstated.)

Here are some preliminary thoughts based mostly, like Amitay's analysis, on the ten highest scoring passcodes. (Since he has kindly shared his data with me, I plan to do a lot more work in the near future on the strategies people employ and on how they might be improved.)

It turns out that 15% of the collected passcodes could be found in the top ten, which consists of the following:

1. 1234
2. 0000
3. 2580
4. 1111
5. 5555
6. 5683
7. 0852
8. 2222
9. 1212
10. 1998

The *iPhone* gives you ten chances to try an activated four-digit screenlock passcode before locking you out. As Amitay et al. have suggested [4, 5], this gives an intruder a disconcertingly good chance of getting in using only the top ten. Other security applications are less forgiving, but selection strategies still bear closer examination.

MNEMONIC LOGIC

The mnemonic logic behind the top ten numbers is more obvious in some cases than in others.

It's hard to think of a more memorable (or obvious) passcode than 1234, for the same reason that 12345 and 123456 regularly appear in password top ten lists – the latter is usually right at the top.

However, any sequence of four *identical* digits is likely to be almost as popular: in this instance, we have 0000, 1111 and 2222 all in the top ten. My guess is that while 3333, 4444 etc. don't feature in the top ten, they're probably not far behind. While 0000 is particularly easy to remember (and therefore to guess), it seems likely that people might choose a different single number according to some rule that makes it more memorable for them, then repeat it as necessary – just as some people use aaaaaaaaa, ggggggggg or zzzzzzzzz or a similar alphabetical sequence for passwords. (However, positioning and accessibility on the keypad may also have a bearing.) Of course, the length of a same-character string may vary according to the requirements of the service demanding the password/passcode: however, that makes very little difference to the ease with which it can be guessed. In the course of a 'dictionary attack' where passwords, passcodes or passphrases are tried according to an ordered list of possibilities, these are likely to be tried very early in the attack.

But why are the other sequences apparently so popular? Figure 1 shows a fairly standard keyboard layout for a basic cellphone/feature phone. Virtual numeric keypads for making phone calls from a smartphone generally follow the same pattern, but have a virtual QWERTY keypad for other kinds of data input, while some feature phones and smartphones have a miniature (hardware) QWERTY keyboard.

Some sequences can be explained by pattern. The middle column of the keypad in descending order gives you 2580, the third most popular choice according to the top ten list.

Figure 1: Standard keyboard layout for a basic cellphone/ feature phone.

Going up the other way – which is just as easy but perhaps a little less intuitive – gives you 0852, the seventh most popular choice.

The middle column is the only one that gives you four digits – the most common length for a PIN – so that probably explains the popularity of these two pattern/code pairs. Other vertical choices in combination with the 0 character are possible, but apparently less popular: 1470 is the 51st most popular choice, while 3690 is the 68th most popular choice. Curiously, given that keyboard patterns are an acknowledged mnemonic aid [6], the reverse patterns did not occur within the sample.

What about 5683? Amitay suggests, convincingly enough, that this is less random than it seems. On the basic phone keypad in Figure 1, you'll see that the letters associated with the number 5683 provide a simple mnemonic using the word LOVE:

(5) JKL (6) MNO (8) TUV (3) DEF

However, that particular association wouldn't work on devices with a QWERTY keyboard like that found on a *BlackBerry* (Figure 2).

Thanks to the single letter-to-number pairing on most *BlackBerry* keypads, there are relatively few four-letter words that conveniently match the nine available letters (as shown in Figure 2: w, e, r, s, d, f, z, x, c). So the single letter-to-number pairing on this type of keypad militates against this particular memorization strategy. The more traditional layout in Figure 1, meanwhile, allows the use of the full alphabet and thus the use of real words and other alphabetical strings as a memory aid, even where passcodes and PINs are limited to four digits.

Figure 2: BlackBerry with QWERTY keyboard.

What about 1212? That would be easy for me to remember, because I'm old enough to remember when New Scotland Yard was the headquarters of London's Metropolitan Police and its very famous telephone number was Whitehall 1212. Later generations might simply use it because a simple [n; n+1; n; n+1] sequence is almost as easy to remember as [n; n; n; n;].

And 1998? Amitay suggests that people use four-digit sequences relating to years that have special significance for them, such as their date of birth or date of graduation. One of the nice things about being my age is that you have a lot of memorable dates behind you – if you can remember them, of course, and are confident they aren't too public to be safe.

What does this tell us about other digital passcodes? Telephone keypads are not always the same as ATM keypads, most significantly in that while even antique rotary telephone dials have letters as well as numbers [7] (though the matching of letters to numbers hasn't always been consistent), not all ATM keypads do. While many modern keypads have the same layout as the telephone keypad shown in Figure 1, some use the common calculator configuration shown in Figure 3:

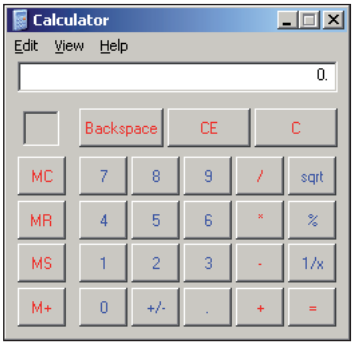


Figure 3: Common calculator configuration.

Many (perhaps even most) ATM keypads and many of the digital safes you find in hotel rooms also use this layout, so the stratagems relating to memorable dates or number sequences are probably also commonly used for ATM PINs.

Some keypads (including numeric keypads for computers and many models of calculator) use almost the same layout but in reverse.

This is a layout that has been used for fast data entry for many years in business. It may not be so frequently encountered in contexts commonly associated with numerical passcodes but nonetheless, it's still worth noting that in contexts where a keypad like this *is* in use, a similar strategy would probably result in the use of 7410 and 0147 rather than 0852 and 2580, and the off-centre positioning of the 0 may further lessen the likelihood of its use in combination with the other columns.

Rasmussen and Rudmin [6] offer a list of mnemonic strategies:

1. Learning by rote
2. Remembering by keypad patterning
3. Code re-use
4. Code with personal meaning
5. Code written down and kept separately
6. Code stored in mobile phone
7. Code concealed in a phone number
8. Numbers paired with letters
9. Written down and kept in proximity
10. Written down but rearranged
11. Notated as a transform of the code.

This data gives an opportunity to confirm to some extent the degree to which these strategies are used. More importantly, perhaps, while articles on the best and worst strategies for choosing passcodes are not in short supply, the data gives us a better starting point for evaluating the entropic efficacy of these strategies as the basis for better recommendations to end-users. And that's a topic I certainly plan to return to.

REFERENCES

[1] Harley, D. Good passwords are no joke. SC Magazine, 2011. http://www.scmagazineus.com/good-passwords-are-no-joke/article/204675/.

[2] Amitay, D. Most Common iPhone Passcodes. 2011. http://amitay.us/blog/files/most_common_iphone_passcodes.php.

[3] Amitay, D. Big Brother Camera Security. 2011. http://amitay.us/projects/big%20brother.php.

[4] Harley, D. Passcodes and Good Practice. Mac Virus, 2011. http://macviruscom.wordpress.com/2011/06/15/passcodes-and-good-practice/.

[5] Cluley, G. The top 10 passcodes you should never use on your iPhone. Naked Security, 2011. http://nakedsecurity.sophos.com/2011/06/14/the-top-10-passcodes-you-should-never-use-on-your-iphone/.

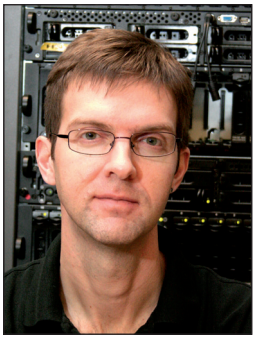
[6] Rasmussen, M.; Rudmin, F. W. The coming PIN code epidemic: A survey study of memory of numeric security codes. Electronic Journal of Applied Psychology. 6(2):5-9 (2010). http://ojs.lib.swin.edu.au/index.php/ejap/article/viewPDFInterstitial/182/220.

[7] http://www.zyra.info/phonedial.htm.

OPINION

STUX IN A RUT: WHY STUXNET IS BORING

John Aycock
University of Calgary, Canada



The much-storied Stuxnet worm is unworthy of the hype surrounding it. The biggest surprise is that Stuxnet contains no surprises, and as such it suggests a general failure of security to respond to threats that are well known. The erroneous characterization of Stuxnet as 'game-changing' does raise other questions, however: what are the hallmarks of real game-changing security events, and why don't we see more of them?

INTRODUCTION

In case you've somehow managed to avoid hearing about Stuxnet, here are the essentials: Stuxnet is a computer worm that has spread and infected machines primarily in Iran. It seems to have targeted uranium enrichment facilities, the output of which can be used in nuclear reactors as well as nuclear weapons; the latter is naturally of some concern to the governments of countries that don't see eye-to-eye with Mahmoud Ahmadinejad. The source of Stuxnet, meanwhile, has been conclusively narrowed down to Planet Earth.

We are accustomed to media reports about malware containing hyperbole and gushing superlatives, but perhaps the biggest thing to note about Stuxnet is that security professionals – not the media – have been describing it using terms usually reserved for reviewing Broadway productions: 'a game-changer', 'a watershed moment', 'I've never seen anything like it'. Stuxnet must be impressive indeed.

There is always one curmudgeonly, contrary Broadway critic, however. That would be me. My view is that Stuxnet is really not that interesting at all, and is at best yet another unfortunate illustration of the fact that the security emperor has no clothes. Let me explain why.

CHANGING GAMES AND SHEDDING WATER

As in history, key events and developments in computer security are best judged in the fullness of time. It is only

then that we have the luxury of hindsight and a surer understanding of the ramifications of a particular event.

If history repeats itself, however, then we can learn some general lessons that can be applied to the present, when hindsight is not an option. That would seem to suggest that there might be some general lessons we can learn about computer security, too. Can we identify a game-changing, watershed event when it occurs?

I propose the following tests. A security threat that meets at least one of the following criteria may be well on its way to becoming a watershed event:

1. Do defences have to be changed in a substantial way to respond to the threat?
2. Does the threat constitute a major shift in motivation for the adversary?
3. Is the adversary using a new business model?

The first test is the most important indicator of a noteworthy threat. Being unable to effectively respond to a threat without a substantive change in defences is a clear sign that something significant has occurred. The latter two tests can be seen as precursors: a change in motivation or modus operandi likely indicates that users are to be targeted in some new ways, that will eventually demand new defences.

Applying these tests, the first game-changer was the computer virus. Not the first known virus in the wild in 1969 [1, 2], not *Apple II* viruses in 1982 [3, 4], and not Fred Cohen's research in 1983 [5], though. The game-changer was the growing glut of PC viruses in the late 1980s – including Brain, Stoned and Jerusalem – that necessitated the development of anti-virus software, a substantial change in defence. The virus is still the embodiment of malicious software to the media and general public.

The next game-changing threat was polymorphism in the early 1990s. Simply put, a polymorphic virus changes its appearance to one of millions or billions of new forms on each new infection. Looking for static signatures is no longer sufficient for detecting polymorphic viruses because it is infeasible to enumerate all possible ways the virus may manifest itself. This led to another major shift in defence, anti-virus emulation, where suspect code is coaxed to run in a safe environment, in the hopes that it reveals any malevolent intentions or (ideally) a unique signature. This nontrivial change in defences involved a 'painful rearchitecting' of anti-virus software [6, p.264].

Moving forward to the mid-1990s, we come to macro viruses. While a proof-of-concept macro virus existed in 1989 [7], they went mainstream in 1995 with the release of Concept – which, ironically, was inadvertently shipped by *Microsoft* on a *Windows 95* compatibility test CD-ROM [8]. A macro virus is a virus written in a macro language, a programming language whose code can be embedded, in this case, in *Microsoft Word* documents. Macro viruses changed the security game because defences could no longer focus solely on executable files. Even data files could now be a threat, and defences had to adjust accordingly.

The second test, a major shift in motivation, leads to the next game-changer: money. Money made by stealing people's data, to be precise. For reference, phishing started in the mid-1990s, and spyware started ramping up through the 2000s. The coming together of four factors set the stage for this shift in motivation. To start with, there had to be a lot of computers connected to the Internet. Then, online banking and online commerce services needed to appear, followed by people using them (simply building a service doesn't guarantee that people will use it – just ask any dotcom startup). Finally, the adversary needed to realize that there was money to be made. Users were no longer just bystanders, owners of the computers on which viruses and worms happened to be spreading. Users and their data became a target; this led to defences such as anti-phishing toolbars and anti-spyware programs.

The final game-changing example is the botnets that started appearing in the early 2000s, compromising computers connected via the Internet that can be controlled by an adversary from afar. The necessary condition was the appearance of a large pool of vulnerable, always-on, always-connected computers. These computers have been repurposed by adversaries, unbeknownst to their owners, for stealing information, sending spam and conducting distributed denials of service. Effective defences now have to look beyond a single computer, beyond a single network, and beyond a single country. Maliciousness scales.

An important observation is that none of the examples above are singular. It is not one virus or one worm or one Trojan horse that is noteworthy by itself. In computer science terms, this makes perfect sense. One thing is a special case, and can be dealt with as a special case; a group of like things, on the other hand, demands a general solution.

HONOURABLE MENTIONS

Some threats looked promising as game-changers, yet didn't quite make the cut.

The Internet worm, aka the Morris worm, is an obvious contender and perhaps the closest parallel to Stuxnet. Released in 1988 by Robert Morris, Jr. (now a faculty

member at MIT), the worm pounded the prehistoric Internet. It was a singular instance of malicious software, and not a watershed moment according to the above criteria. If anything, the worm was an indictment of programming and system administration practices, but these poor practices were not news.

In *Hamlet*, Shakespeare wrote 'the lady doth protest too much,' and it is interesting to note that the tenor of some worm analyses was not only technical, but at times laced with disdain and derision; Spafford belittling the worm author's programming skills comes to mind [9]. Certainly the worm did cause disruption, but the security community was caught with its proverbial pants down, and it's not inconceivable that some psychological projection was in play. The Cornell Commission investigating the worm found that 'At least one of the security flaws exploited by the worm was previously known by a number of individuals, as was the methodology exploited by other flaws' [10, p.707]. In fact, the buffer overflow technique used by the worm was known as far back as the Anderson report in 1972, 16 years previously [11]. The Internet worm was not a turning point, it was an embarrassment.

The causal chain of other potential game-changing events is too long to claim any real impact. The inclusion of a TCP/IP stack in *Windows 95*, for example, went a long way towards creating a large pool of vulnerable machines. It was hardly a security threat in itself, though. And where is the line drawn – could the finger not be pointed as easily at the development of the Arpanet? It's a slippery slope.

Some would-be security game-changers are simply too early in their lifecycle. Cellular (smart) phones and social networking have undeniably transformed our lives, yet there are no radical, widespread new security threats from them. There are threats, yes, but for the time being they are old threats, repackaged for new platforms. Unfortunately, this is bound to change. The smartphone as an e-wallet, for example, dangles a tantalizing target for adversaries.

HOW STUXNET STACKS UP

Stuxnet doesn't fare well as a game-changer when reason, rather than rhetoric, is used. There are eight key features of Stuxnet (see the analysis in [12]):

- Stuxnet is large and complicated.

  If being a large and complex piece of software was a valid criterion, then every release of *Microsoft Office* would be a game-changer.

- Stuxnet is a targeted attack.

  Targeted attacks have been a concern for years – long, long before Stuxnet appeared.

- Stuxnet spreads in multiple ways.

  Again, Stuxnet is unoriginal, as a web search for 'multipartite virus' will attest.

- Stuxnet targets industrial control systems.

  The poor security of industrial systems comes as no surprise, and in fact Stuxnet isn't the first example of an attack against them (the 2000 sewage incident in Australia is a good example [13]).

- Stuxnet uses multiple exploits, some of which are zero-day.

  Stuxnet isn't the first threat to use zero-day exploits, and having several of them deserves some kudos but does not require any different defence.

- Stuxnet contains rootkits to hide itself.

  While having two rootkits (one for *Windows*, one for the programmable logic controller) is an interesting idea, rootkits themselves aren't new.

- Stuxnet misused code-signing certificates.

  This would be most distressing if certificates hadn't been abused or wrongly issued before (they have), and if signed code really provided the safety and security guarantees that people want (it doesn't).

- Stuxnet's motivation was espionage and/or sabotage.

  This is not a new motivation for malicious software. I would argue that Stuxnet could be considered a failure in some sense, because it was discovered (with the possible exception of Bond-esque antics, covert operations ideally remain surreptitious).

While Stuxnet is impressive in terms of the effort and investment it took to create, it is just not the dawning of a new age. There is nothing new to see here, there is nothing that security professionals haven't seen before. Especially telling is that fact that so much of the Stuxnet hullabaloo is focused on its analysis, ignoring the fact that, once its existence was known, anti-virus products could pick it off without difficulty. Or, as is phrased in *Symantec*'s threat assessment of Stuxnet, 'Removal: Easy' [14].

MALICIOUS MONOTONY

As a long-time observer of malicious software trends, the lack of game-changers is disappointing, in a perverse way. It is also understandable: the adversary is now typically a

businessperson, whose goal is to make money. If this goal is being met, i.e. enough malicious software is making it past anti-malware defences, then that's sufficient.

One example is the rapid repacking of malicious software. A packer is a relatively small investment, and can be bought from a third party then used to evade defences by overwhelming them with 'new' (or at least new-looking) samples of what is actually the same malicious software.

In contrast, Stuxnet is a bad investment. In general, there is no business case for one-shot, complicated, novel malicious software. The only exception is where a widely deployed defence is too good. Then, and only then, does the business-oriented adversary have the incentive to innovate. This back-and-forth was seen clearly with early spam and anti-spam, and arguably the success of early anti-virus was an impetus for polymorphic code.

This implies that security is not only risk management. 'Good' defences are the ones that keep adversaries in a sweet spot, where the adversary succeeds enough to be satisfied but doesn't fail enough to evolve. It's a strange notion, that losing the security game once in a while might be necessary to strike a healthy balance overall.

It could also be argued that malware like Stuxnet and the Morris worm has an educational value. Thanks to them, the issues are now in the public eye; they are prominent examples that can be used to justify funding for security, designing security into systems and additional security measures. But the sword slashing into popular consciousness is double-edged. Overwhelming evidence from the last decade suggests that hysterics over large-scale security events may lead to an unnatural obsession with the last attack, rather than promoting activity of any real benefit.

The education argument is fair, however, in that a game-changer must be considered relative to a particular audience. To the public, Stuxnet is a game-changer, without a doubt. To the security professionals charged with safeguarding everything from smartphones to critical infrastructure, Stuxnet and its successors should have been imagined long ago. Stuxnet should be little more than validation of security professionals' fears, not a surprise.

ACKNOWLEDGEMENTS

An early version of this paper was presented at the University of Glasgow as a SICSA Distinguished Visitor Talk. The author's work is supported in part by the Natural Sciences and Engineering Research Council of Canada. Thanks to Darcy Grant, Mike Locasto and Tim Storer for their insightful comments. The names of the security professionals mentioned in the introduction have been purposely omitted.

REFERENCES

[1] Benford, G. Worlds Vast and Various. EOS, 2000.

[2] Benford, G. Catch me if you can. Commun. ACM, 54(3):112-111, 2011.

[3] Dellinger, J. Re: Prize for most useful computer virus. Risks Digest, 12(30), 1991.

[4] Skrenta, R. Elk cloner. http://www.skrenta.com/cloner.

[5] Cohen, F. Computer viruses: Theory and experiments. Computers & Security, 6(1):22-35, 1987.

[6] Ször, P. The Art of Computer Virus Research and Defense. Addison-Wesley, 2005.

[7] Highland, H. J. A macro virus. Computers & Security, 8(3):178-188, 1989.

[8] WinWord.Concept. Virus Bulletin, October 1995, p.3. http://www.virusbtn.com/pdf/magazine/1995/199510.pdf.

[9] Spafford, E. H. The Internet worm program: An analysis. Technical Report CSD-TR-823, Purdue University, Department of Computer Sciences, 1988.

[10] Eisenberg, T.; Gries, D.; Hartmanis, J.; Holcomb, D.; Lynn, M. S.; Santoro, T. The Cornell commission: On Morris and the worm. Commun. ACM, 32(6):706-709, 1989.

[11] Anderson, J. P. Computer security technology planning study: Volume II, Oct. 1972. ESD-TR-73-51, Vol. II.

[12] Falliere, N.; O'Murchu, L.; Chien, E. W32. Stuxnet dossier (version 1.4). Symantec, February 2011. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf.

[13] Slay, J.; Miller, M. Lessons learned from the Maroochy Water breach. In Goetz and Shenoi, eds, Critical Infrastructure Protection, pp.73–82. Springer, 2008.

[14] Shearer, J. W32.Stuxnet. Symantec Security Response, 2010. http://www.symantec.com/security_response/writeup.jsp?docid=2010-071400-3123-99.

END NOTES & NEWS

The 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS 2011) will be held in Perth, Australia 1–2 September, 2011. See http://ceas2011.debii.edu.au/.

(ISC)² Security Congress takes place 19–22 September 2011 in Orlando, FL, USA. The first annual (ISC)² Security Congress offers education to all levels of information security professionals, not just (ISC)² members. For more information visit http://www.isc2.org/congress2011.

Cairo Security Camp takes place 30 September to 1 October 2011 in Cairo, Egypt. This annual event targets the information security community of the Middle East and North Africa. IT professionals and security practitioners from throughout the region are invited to attend. See http://www.bluekaizen.org/cscamp.html.

VB2011 takes place 5–7 October 2011 in Barcelona, Spain. The organizers are now seeking submissions for 'last-minute' technical papers dealing with up-to-the-minute specialist topics (deadline for submissions 8 September). For full details and online registration see http://www.virusbtn.com/conference/vb2011/.

RSA Europe 2011 will be held 11–13 October 2011 in London, UK. For more information see http://www.rsaconference.com/2011/europe/index.htm.

The MAAWG 23rd General Meeting takes place 24–27 October 2011 in Paris, France. See http://www.maawg.org/.

The Hacker Halted Conference takes place 25–27 October 2011 in Miami, FL, USA. The conference is preceded by the Hacker Halted Academy (a range of technical training and certification classes) 21–24 October. For more information about both events see http://www.hackerhalted.com/2011/.

The CSI 2011 Annual Conference will be held 6–11 November 2011 in Washington D.C., USA. See http://www.CSIannual.com/.

The sixth annual APWG eCrime Researchers Summit will be held 7–9 November 2011 in San Diego, CA, USA. The summit will bring together academic researchers, security practitioners and law enforcement to discuss all aspects of electronic crime and ways to combat it. For more details see http://www.antiphishing.org/ecrimeresearch/2011/cfp.html.

The 14th AVAR Conference (AVAR2011) and international festival of IT Security will be held 9–11 November 2011 in Hong Kong. For details see http://aavar.org/avar2011/.

Ruxcon takes place 19–20 November 2011 in Melbourne, Australia. The conference is a mixture of live presentations, activities and demonstrations presented by security experts from the Aus-Pacific region and invited guests from around the world. For more information see http://www.ruxcon.org.au/.

Takedowncon 2 – Mobile and Wireless Security will be held 2–7 December 2011 in Las Vegas, NV, USA. EC-Council's new technical IT security conference series aims to bring industry professionals together to promote knowledge sharing, collaboration and social networking. See http://www.takedowncon.com/ for more details.

Black Hat Abu Dhabi takes place 12–15 December 2011 in Abu Dhabi. Registration for the event is now open. For full details see http://www.blackhat.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/