TÉVHITEK ÉS FÉLREÉRTÉSEK AZ ANTIVIRUS PROGRAMOKKAL KAPCSOLATBAN

ZERO IT LAB

Introduction

- CTO @ Zero
- Member @ Crysys Lab & C0r3dump
- RE is love, RE is life
- Opening expensive calculators at times
- <3 capturing the flag

zero IT Lab

Cybersecurity challenges and risks

- Risk = Likeliness * Impact
- Types of risks to personal/corporate security?
  - Physical breach
  - Social engineering
  - Operational security (opsec)
  - Technological vulnerability (in software/hardware)

Vulnerabilities in modern software (standalone, not a web app)

- If written in a memory unsafe language, memory corruption can occur
- If written in a language where the programmer has to handle memory (e. g. malloc()/free()), memory corruption can occur (use after free)
- Logic bugs (harder to generalize)

How do we solve memory corruption?

- Just use a memory safe language lool (/s)
- Memory safe languages tend to produce less efficient machine code (runs slower)
  - lot of (legacy) code already written in memory unsafe languages
  - numerous operating environments require efficient machine code (and by implication, memory unsafe languages)

Portrayal of post-exploitation tools

- Focus on the “malicious code” (aka a “virus”) instead of the vulnerability / source of arbitrary code execution
  - Common theme among mainstream media & general discussions
- Antivirus / Endpoint protection software is advertized as a solution
- Lot of outsiders pick up on this trend unknowingly, unwillingly and without understanding

Example given: WannaCry

- Countless news pieces follow the said pattern
- Even wikipedia starts with: “The WannaCry ransomware attack was a May 2017 worldwide cyberattack by the WannaCry ransomware cryptoworm,”

ZERO IT LAB

Example given: WannaCry

- Who is going to talk about …
- ETERNALBLUE ??
- How the NSA ignorantly stockpiles exploits as a habit ??
- Why we don't have proper open fuzzing and sanitizers around certain windows components?

zero IT Lab

Vulnerabilities/Exploits > Post-exploitation tools

- Malware researchers and Antivirus software (AV) both care about post-exploitation tools (binaries) & frameworks (e. g. Trojans/RATs)
  - (IMO) Not completely useless but fundamentally wrong
- The way ***arbitrary code execution*** is achieved is what is relevant for good defense

Vulnerabilities/Exploits > Post-exploitation tools

Tavis Ormandy

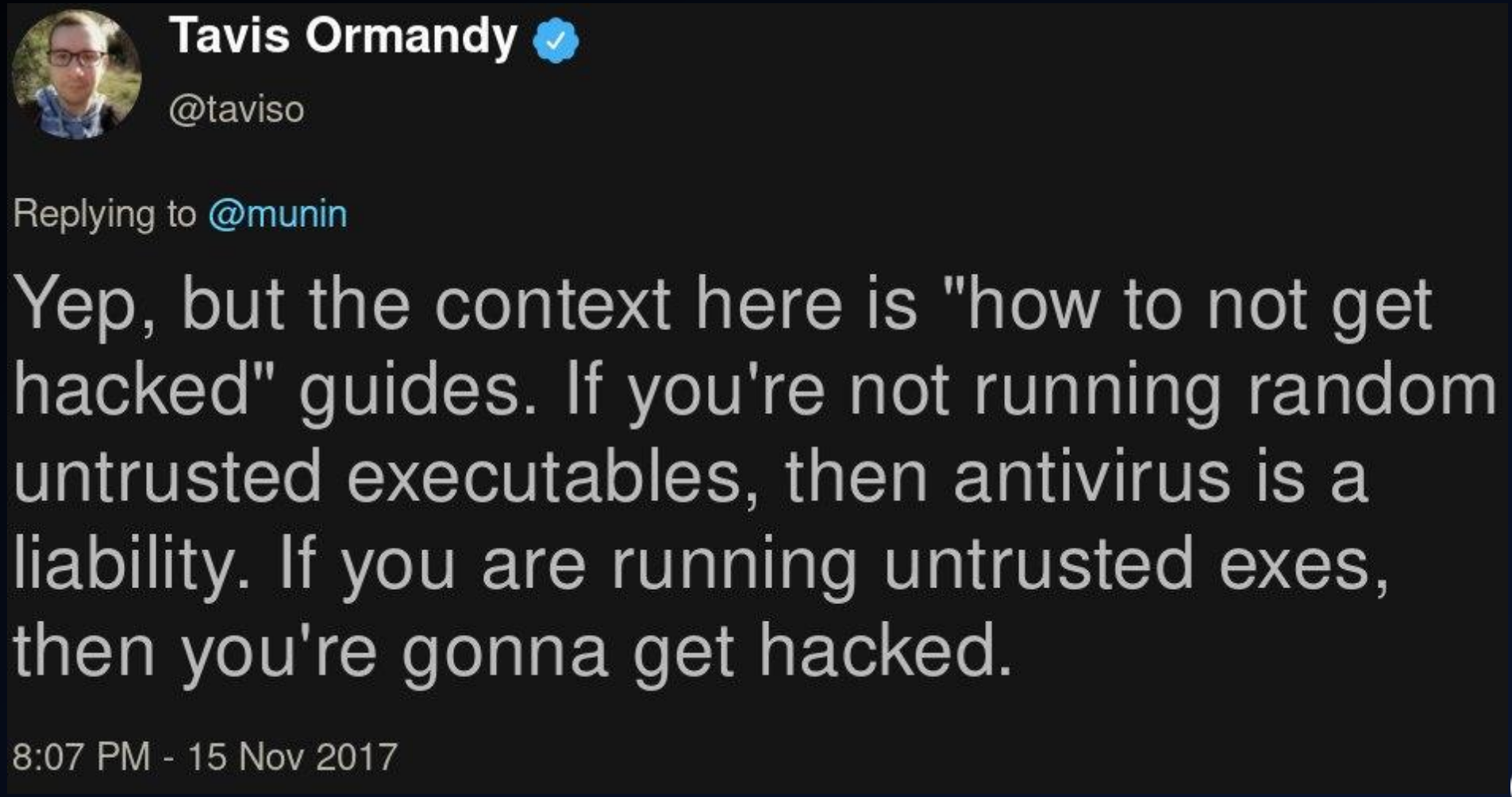
@taviso

Replying to @munin

If you insist on running untrusted executables and refuse to fix that problem, then you're screwed and antivirus doesn't change that. It's plausible AV will increase the amount of time your system is usable between reinstalls, but that is not security.

7:56 PM - 15 Nov 2017

Vulnerabilities/Exploits > Post-exploitation tools



Vulnerabilities/Exploits > Post-exploitation tools

Tavis Ormandy

@taviso

Replying to @munin

I don't think it's as complicated as people claim. People understand how to evaluate if they would let you use their laptop in person if you asked. They don't understand that running an exe you wrote is exactly equivalent. IMO, when you explain it like this people get it.

8:12 PM - 15 Nov 2017

Free software

- 0: Run as you wish, for any purpose
- 1: Ability to study and change the program as you wish
- 2: Freedom to redistribute as you wish
- 3: Freedom to redistribute your changes

http://www.gnu.org/philosophy/free-sw.en.html

Freedom? Security? Privacy?

- Related? ...
  - Privacy is part of freedom?
  - Surveillance silences dissenting views!
  - Relience on security to protect privacy (and vica versa)
- So what about freedom and security?
  - Security by obscurity is bs! Hence proprietary != more secure
- Interdependent!

https://puri.sm/posts/why-freedom-is-essential-to-security-and-privacy/

Freedom = Ownership

- Free software allows you to own your system and find memory corruption, on equal footing!
  - run Sanitizers like Google's Address Sanitizer or Thread Sanitizer (https://github.com/google/sanitizers)
  - run extensive (and efficient!) fuzzing using binary instrumentation (that requires recompilation)
  - patch and deploy fixes to vulnerabilities at your own pace

https://puri.sm/posts/why-freedom-is-essential-to-security-and-privacy/

What happens with antivirus instead?

- AV software is almost always:
  - written in a memory unsafe language, typically C/C++
  - proprietary (meaning: no oversight, no sanitizers, limited fuzzing)
  - users are dependent on the AV vendors to fix vulnerabilities in AV software
  - runs highly privileged (ring0) custom components via a kernel driver

Memory unsafety in AV software

- AV software implements a lot of complex functionality (more code -> more vulnerabilities on average)
- The functionality implemented is a great attack surface for interactionless attacks (automatic scans, actions from AV)
- Multi-stage attack payloads are easy deliver
- Code execution in the highly privileged component = fully owning the system (violates normal privilege boundaries)

AV software is almost always proprietary

- Other than a few notable exceptions (such as ClamAV); AV software is proprietary
- Even intentional backdoors are hard to discover for anyone other than the limited team of people that work on it
- Using any of these products = blindly trusting the vendor

Real world example #1 - 2016 - Google Project Zero

- How to Compromise the Enterprise Endpoint
- Many **wormable** remote code execution flaws in
  - All Symantec and Norton branded antivirus products, including Symantec Endpoint Protection, Email Security, Protection Engine, Protection for SharePoint servers, etc..
- These products decided that it was a good idea to unpack malware in their kernel driver
- Unpackers and emulators were found vulnerable in products from Comodo, ESET, Kaspersky, Fireeye, and many more..

zero IT Lab

Real world example #1 - CVE-2016-2208

- Stupid beyond belief.. (all values are attacker controlled)

Effectively, we can get Symantec to execute a sequence like this:

```
char *buf = malloc(SizeOfImage);

memcpy(&buf[DataSection->VirtualAddress],
       DataSection->PointerToRawData,
       SectionSizeOnDisk);
```

- aaaand.. this happens in their kernel driver :””D

Real world example #1 - CVE-2016-2208

- Because Symantec uses a filter driver to intercept all system I/O, just emailing a file to a victim or sending them a link to an exploit is enough to trigger it - the victim does not need to open the file or interact with it in anyway.

Effectively, we can get Symantec to execute a sequence like this:

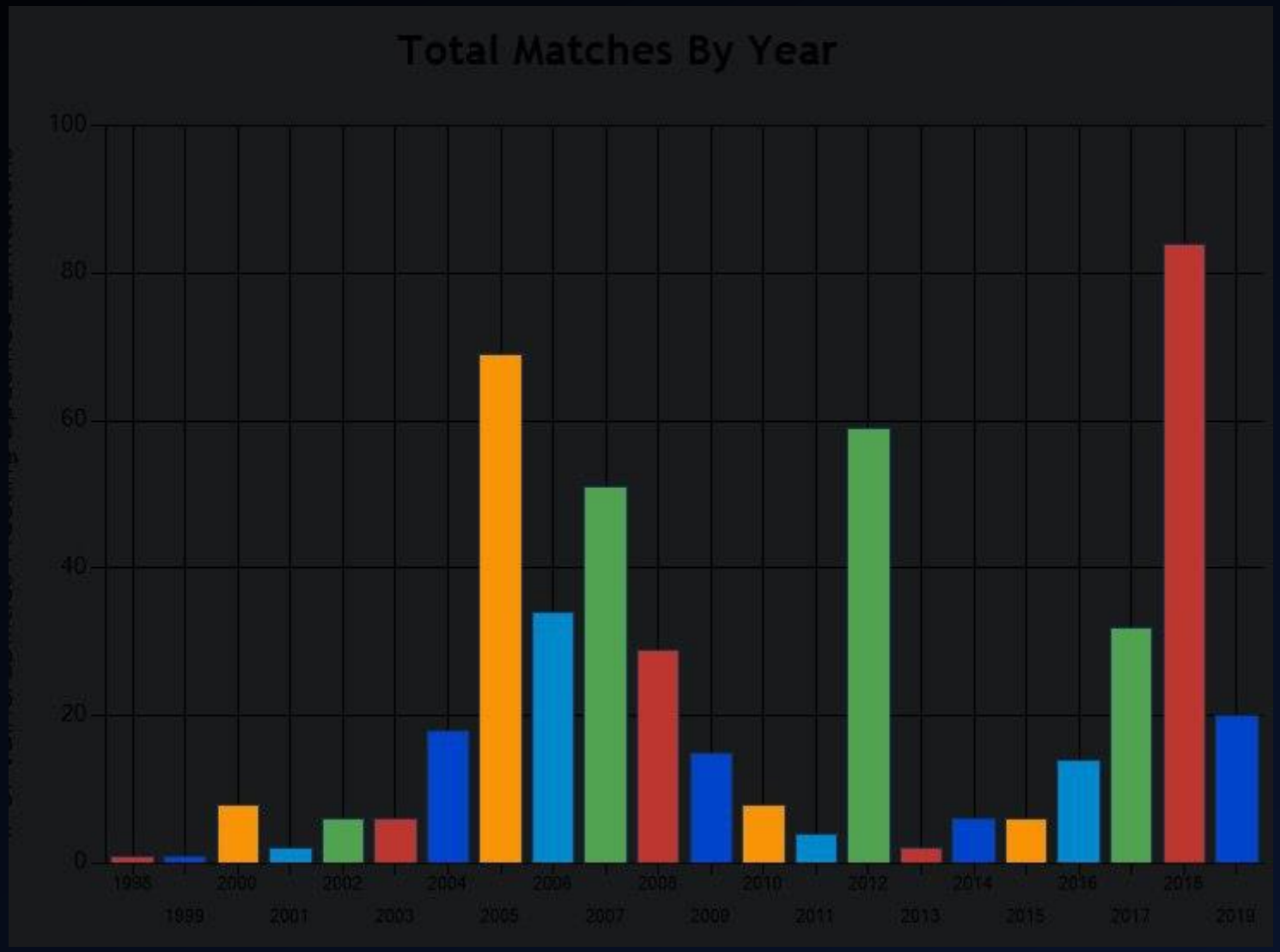
```
char *buf = malloc(SizeOfImage);

memcpy(&buf[DataSection->VirtualAddress],
       DataSection->PointerToRawData,
       SectionSizeOnDisk);
```

Real world example #2 - Comodo AV (Recent)

- CVE-2019-3969: Local Privilege Escalation (CmdAgent.exe)
- CVE-2019-3970: Arbitrary File Write
- CVE-2019-3971: Denial of Service (CmdVirth.exe)
- CVE-2019-3972: Out-of-bounds Read (CmdAgent.exe)
- CVE-2019-3973: Out-of-Bounds Write (Cmdguard.sys)

How common is this exactly?



>80 reported vulnerabilities in nist.gov just last year

ZERO IT LAB

“but mah users are unable to behave correctly”

- Often said in enterprise environments for attacks like phishing, social engineering, etc..
- I challenge you to think about this, how much of this is user naivety VS bad patterns enabled by software?
- Think about signed packages from a package manager on Linux/BSD based systems VS installing software by running random exe-s from internet sites..

In conclusion

- (IMO) AV does not have a place in IT security; it is not offering any solution to any of the fundemental problems we are facing
- You should not be using or forcing people to use proprietary software that often makes them provably less secure
- You should not be running untrusted binary executables.