Into the Unknown: How to Detect BIOS-level Attackers

Xeno Kovah
Corey Kallenberg
John Butterworth
Sam Cornwell

MITRE

Research Team – Credit Where Credit's Due

Corey Kallenberg

John Butterworth

Sam Cornwell

Bob Heinemann

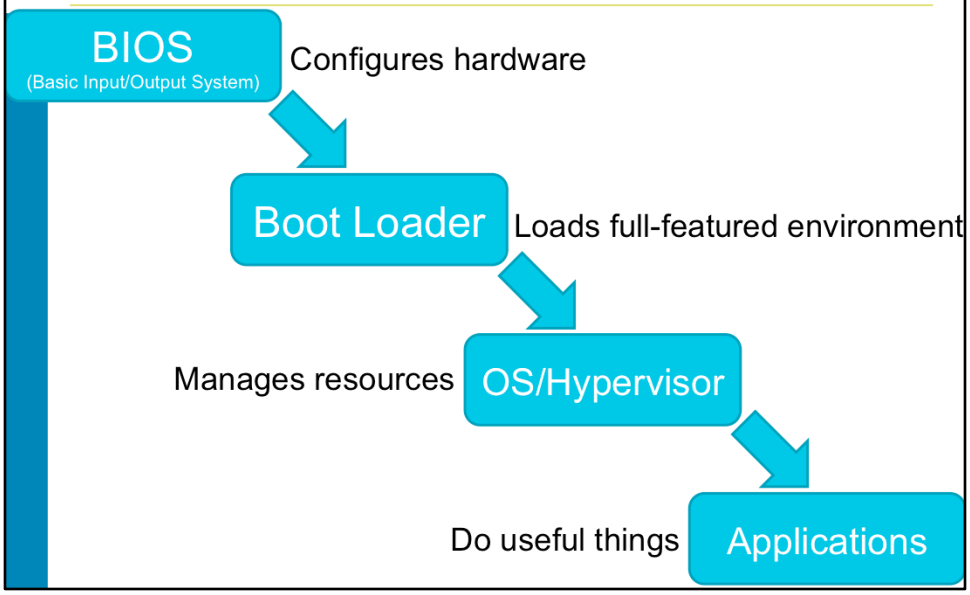
MITRE

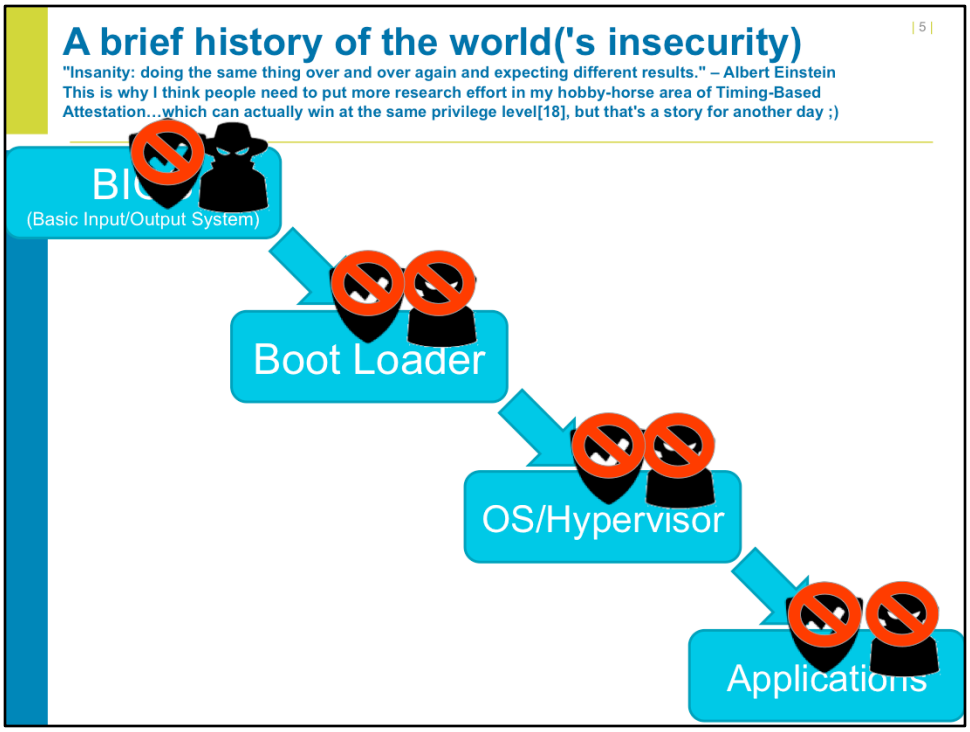
Introduction

- **Who we are:**
  - Trusted Computing and firmware security researchers at The MITRE Corporation
- **What MITRE is:**
  - A not-for-profit company that runs six US Government "Federally Funded Research & Development Centers" (FFRDCs) dedicated to working in the public interest
  - Technical lead for a number of standards and structured data exchange formats such as *CVE*, CWE, OVAL, CAPEC, STIX, TAXII, etc
  - The first .org, !(.mil | .gov | .com | .edu | .net), on the ARPANET

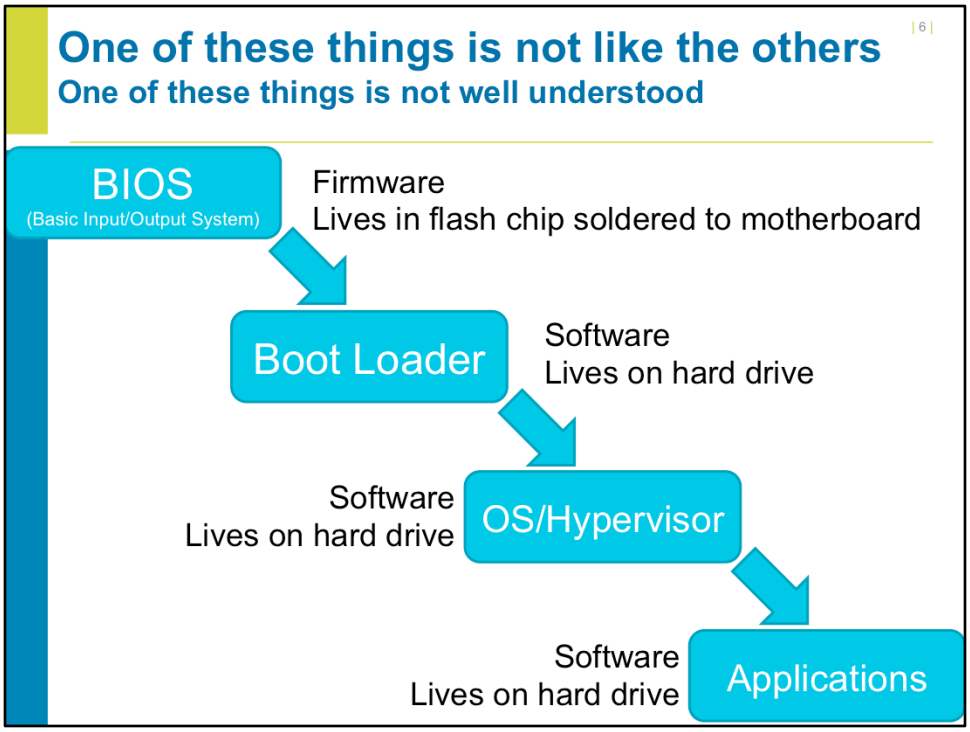
MITRE

How computers do useful things

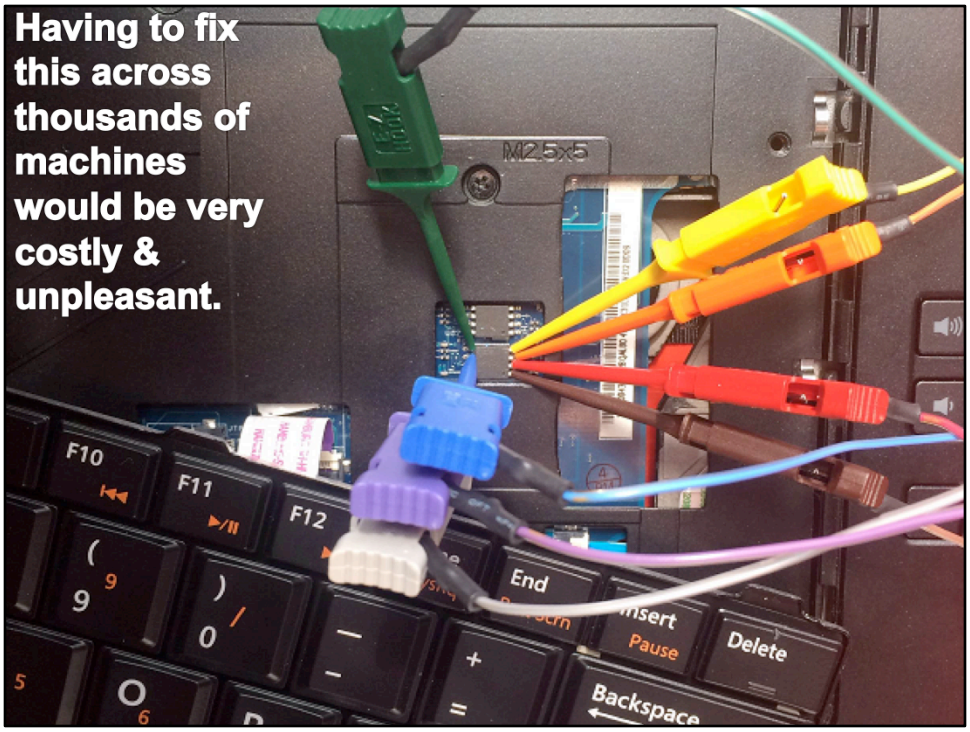




http://icons.iconarchive.com/icons/hopstarter/malware/icons-390.jpg
http://www.iconarchive.com/show/windows-8-icons-by-icons8/Security-Security-Checked-icon.html



The point is, people understand how to recover from other problems. They wipe/ replace the hard drive and reinstall software. They don't know how to recover from firmware attacks.



This is the best possible case, where the BIOS is exposed for easy reflash.

What you don't know *can* hurt you

- **BIOS is the first code that the CPU ever runs.**
  - It can affect and compromise all subsequent code that runs
  - It is a black box that almost no one understands
- **Therefore we needed to become BIOS SMEs and share our knowledge and findings with others**

MITRE

Highlights of what we have found in our short time working on firmware security

- **There were no public tools to confirm BIOS access controls were set properly**
  - And public tools to even *read* the BIOS were spotty at best!
  - So we made one, "Copernicus", and made the binary freely available so anyone could check their system [26]
- **A key Trusted Computing Group technology that supported a secure boot up (the Static Core Root of Trust for Measurement) was weak[18]. But we could strengthen it with our previous work [19]**
- **We found, disclosed, and saw patched the second ever publicly talked about BIOS exploit [13]**
  - Patched by Dell 7/2013, affected 22 Legacy BIOS models CVE-2013-3582, CERT VU# 912156

MITRE

Highlights of what we have found in our short time working on firmware security

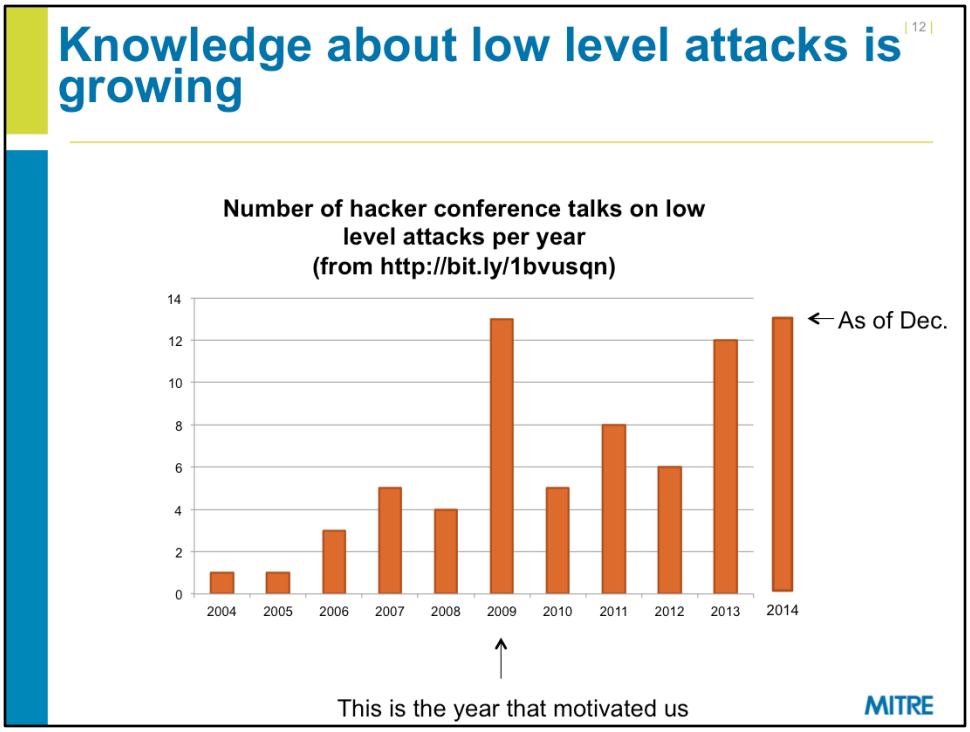
- **Discovered a new type of Man in the Middle (MitM) attack that could universally hide from software-based BIOS integrity checkers**
  - "SMM MitM" attacker dubbed "Smite'em the Stealthy" [27]
  - We made "Copernicus 2" using Intel Trusted Execution Technology to combat Smite'em [28]
- **Problems with Unified Extensible Firmware Interface (UEFI) variables that could lead to bypassing Windows 8 SecureBoot [29]**
  - Co-discovered with Intel

MITRE

Highlights of what we have found in our short time working on firmware security

- **2 confirmed-exploitable buffer overflows in the open source Intel reference UEFI BIOS implementation [31]**
  - CVE-2014-4859 & CVE-2014-4860, CERT VU # 552286
  - This reference code is used by many other OEMs
  - Patched by Intel, Phoenix, AMI, HP (33 enterprise & > 470 consumer models), Dell (39 enterprise models), Lenovo (vuln, TBD models)
  - Waiting for ACK/NACK and/or patches from other vendors
    - Most likely to be silent if ever
  - Insyde says they're not vulnerable
- **And more things still under disclosure moratorium**

MITRE



"We got interested in 2009, but it took us a couple years to shift from our focus on Windows kernel security to finding meaningful new firmware problems & solutions." This year or next will be the high watermark, surpassing 2009's spike of interest.

So What?

- What are some consequences of firmware attacks?

MITRE

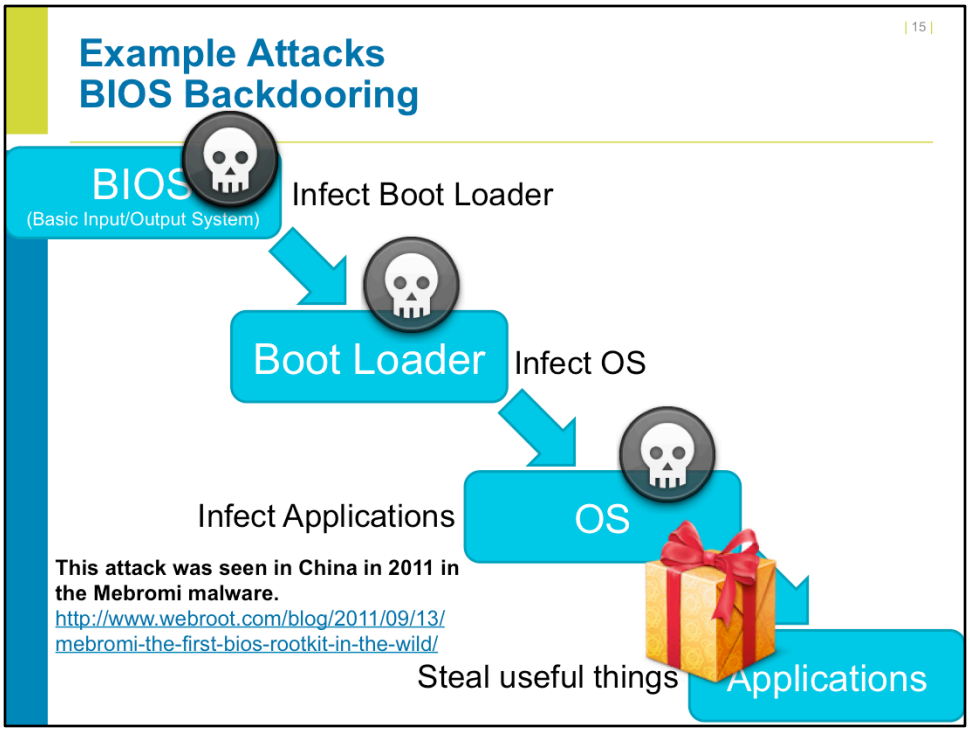
Example Attacks
BIOS "Bricking"

Firmware is corrupted (1 byte is all that's needed)
System will not boot

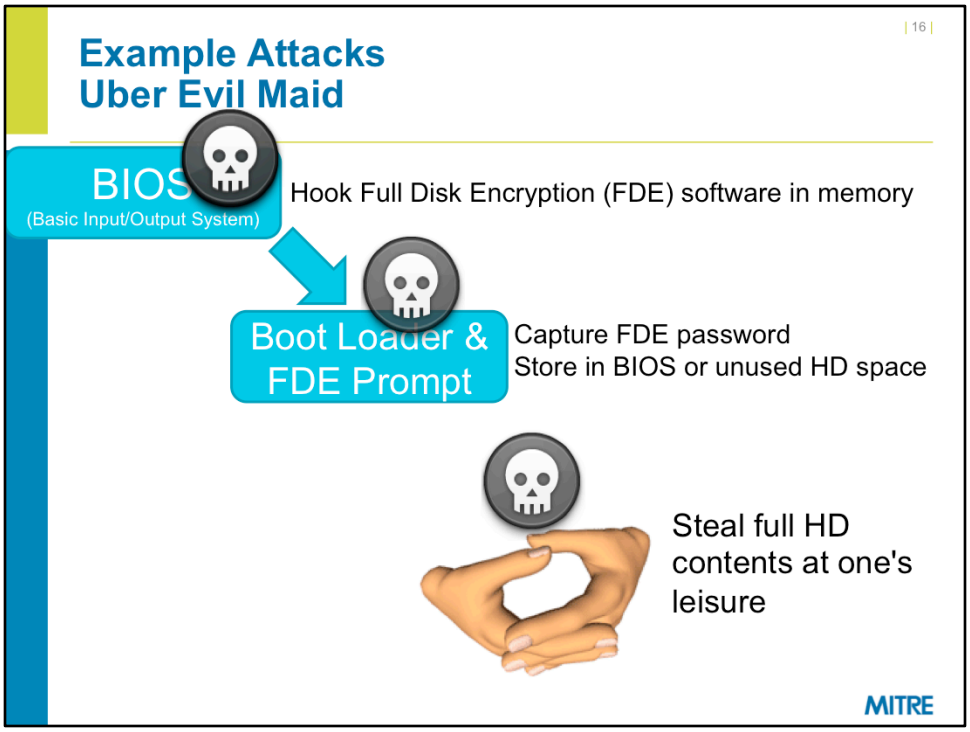
The CIH virus did this as a time-bomb attack on (*supposedly* 60 million) computers in 1998
http://en.wikipedia.org/wiki/CIH_(computer_virus)

MITRE

http://www.wickes.co.uk/content/ebiz/wickes/invt/213633/Facing-Brick_large.jpg

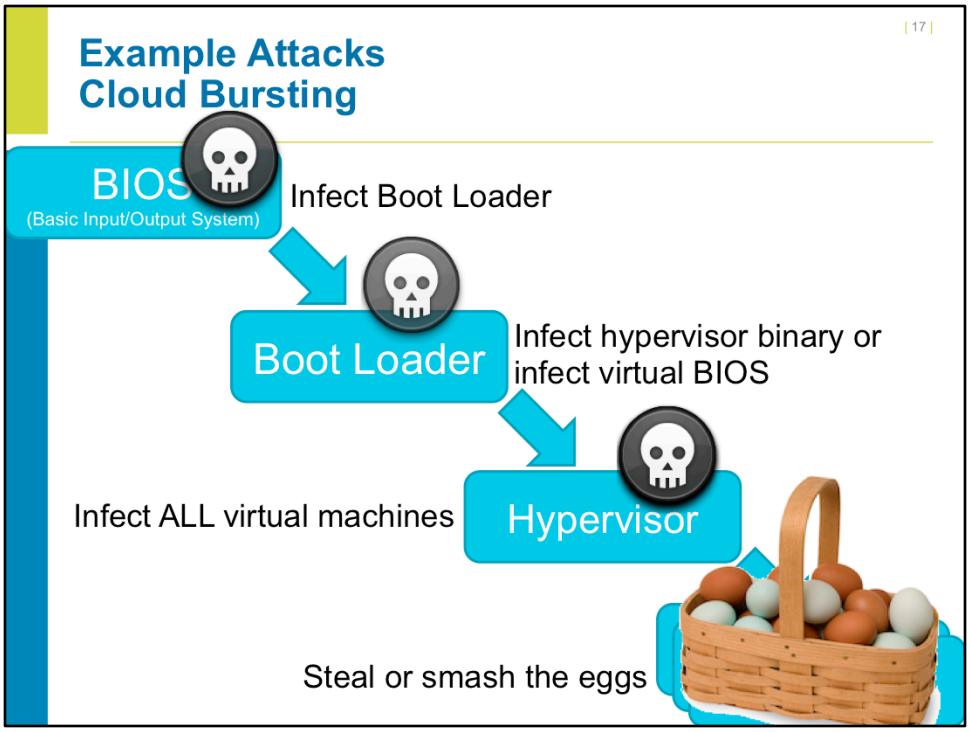


http://www.veryicon.com/icons/system/agua-stacks/evil-stack.html
https://cdn1.iconfinder.com/data/icons/Gifts/512/box1.png



"It's one thing for someone to say what's clearly architecturally possible, and it's another to show a video of specific software being compromised or security software being bypassed."

http://2.bp.blogspot.com/-BtStmGpZOts/TsCIMLUXvwI/AAAAAAAACAc/N19iHf_pGms/s1600/hands_twiddling_thumbs_fast_lg_nwm.gif



http://peteandgerrys.com/tag/pete-gerrys-heirloom-eggs/
http://peteandgerrys.com/wp-content/uploads/2012/04/Combo-in-Basket.jpg

Yeah, but…

- **Building a BIOS level implant is 00b3r-1ee7 and only the lCy g0dz could ever pull that off…and even if they did, it would be so device-specific they would have a hard time getting it to work in the target environment**

  - It took me around 2 weeks to create a PoC maximally capable backdoor that could be delivered via a PoC BIOS exploit for a conference
  - Open source UEFI reference BIOS implementation and well-defined specification is a double edged sword that works for and against defenders

MITRE

https://hbslp.files.wordpress.com/2013/04/dwight-schrute-false.jpg

Marvel Comics
Fantastic Four #13, 1963

Presenting the first appearance of The Watcher!

MITRE

The Watcher

- **The Watcher lives in SMM (where you can't look for him)**
- **It has no built-in capability except to scan memory for a magic signature**
- **If it finds the signature, it treats the data immediately after the signature as code to be executed**
- **In this way the Watcher performs *arbitrary code execution* on behalf of some controller, and is *completely OS independent***

- **A controller is responsible for placing into memory payloads for The Watcher to find**
- **These payloads can make their way into memory through any means**
  - Could be sent in a network packet which is never even processed by the OS
  - Could be embedded somewhere as non-rendering data in a document
  - Could be generated on the fly by some malicious javascript that's pushed out through an advertisement network
  - Could be pulled down by a low-privilege normal-looking dropper
  - Use your imagination

MITRE

The Watcher, watching

Design tradeoffs:

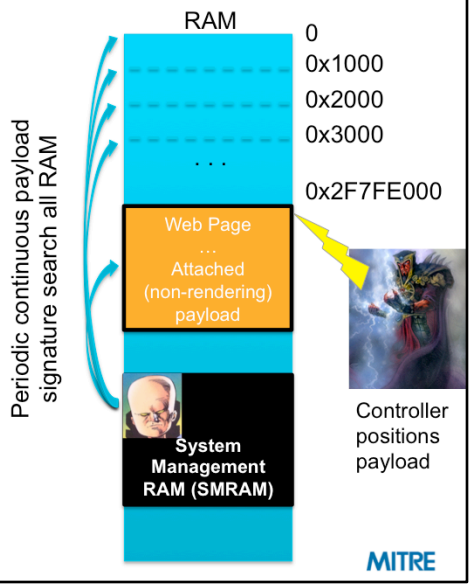
We don't want to scan every 4 byte chunk of memory. So instead we scan every 0x1000-aligned page boundary.

How do we guarantee a payload will be found on a page-aligned boundary?

a) Another agent puts it there

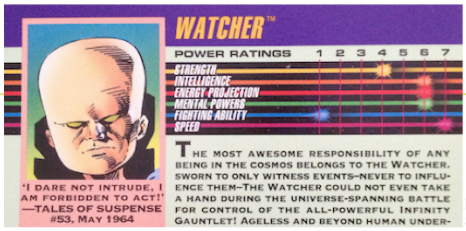
b) Controller prefixes the payload with a full 0x1000 worth of signatures and pointers to the code to be executed (this guarantees a signature will always be found at the boundary or boundary+4)

There are obviously many different ways it could be built.



MITRE

Watcher Stats



Impel 1992
Marvel Universe
Series 2

- A week to get dev env set up (I didn't have my SPI programmer) and to find where to insert the code into SMM so it got called on every SMI
- A week to get it implantable via PoC BIOS exploit
- 2 days to write Watcher + basic print payload
- Watcher itself: ~ 60 lines of mixed C and inline assembly
- Print payload: 35 bytes + string, 12 instructions
- Ultimate Nullifier payload: 37 bytes, 11 instructions

- Overall point: very simple, very small, very powerful
- How likely do you think it is that there aren't already Watchers watching?
- But we can't know until people start integrity checking their BIOSes

MITRE

One Stealth Malware Taxonomy

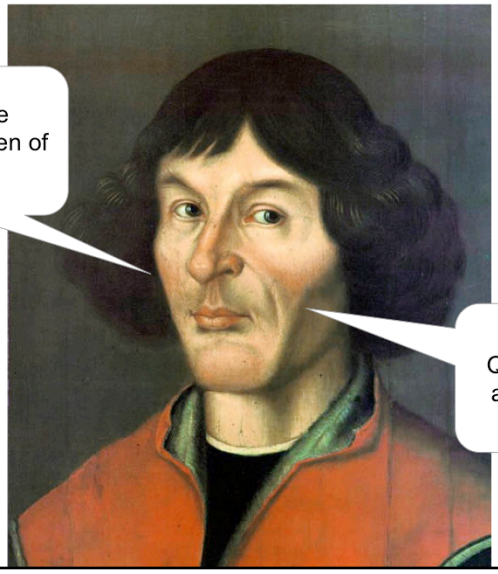
aka "Why would someone bother with a firmware attack?"
(answer: maximum power)

- Ring 3 – Userspace-Based
- Ring 0 – Kernel-Based
- "Ring -1" – Virtualization-Based
  - Intel VT-x(Virtualization Technology for x86), AMD-V (AMD Virtualization), Hypervisor subverted
- "Ring -1.5?" - Post-BIOS, Pre OS/VMM
  - e.g. Master Boot Record (MBR) "bootkit"
  - Peripherals with DMA(Direct Memory Access) (this can be ring 0, -1, or -1.5 depending on whether VT-d is being used)
  - Not a generally acknowledged "ring", but the place I think it fits best
- "Ring -2" – System Management Mode (SMM)
- "Ring -2.25 – SMM/SMI Transfer Monitor (STM)
  - A hypervisor dedicated to virtualizing SMM
  - Another one of my made up "rings", I just added this ring for this presentation :)
- "Ring -2.5" - BIOS (Basic Input Output System), EFI (Extensible Firmware Interface)
  - because they are the first code to execute *on the CPU* and they control what gets loaded into SMM
  - Not a generally acknowledged "ring", but the place I think it fits best
- "Ring -3" – Chipset Based - *probably not valid anymore on modern architectures*
  - Intel AMT(Active Management Technology)
  - Could maybe be argued that any off-CPU, DMA-capable peripherals live at this level?

But BIOS could use VT-d to prevent DMA, and it initializes peripherals, so…?
Yeah, things get squishy at the bottom with non-real-rings.

23

Copernicus Inside



MITRE

Simple & Small: 2 Capabilities

- **BIOS-writability report**
  - "Are we vulnerable to attack?"
  - Indicates whether your BIOS access controls are not properly set, and therefore the systems can be trivially bricked or backdoored
- **Integrity report**
  - "Have we already been attacked?"
  - Dump BIOS flash chip, compare against all the other machines of the same Vendor/Model/Revision, or compare against a single known good

MITRE

BIOS/SMRAM Writability Analysis Demo

- **protections.py**
- http://youtu.be/wVulh2ADsT4

```
C:\copernicus>python protections.py per-version csv
COUNT  BIOS_VENDOR PRODUCT_NAME    BIOS_VERSION SMRAM_UNLOCKED BIOS_UNLOCKED
2      Dell Inc.   Latitude E6430 A11          0              0
1      Dell Inc.   Latitude E6430 A12          0              0

3 CSV files successfully processed
0 (0.0%) CSV files showing unlocked SMRAM
0 (0.0%) CSV files showing unlocked BIOS
0 (out of 0 -- 0.0%) CSV files showing vulnerability to CERT VU#912156
0 (0.0%) CSV files showing vulnerability to CERT VU#255726
0 (0.0%) CSV files showing SMI_LOCK not set
```

0/1, no/yes, can someone easily escalate from ring 0 to SMM, or BIOS?

MITRE

Control flow is king!

- **Control flow starts from a known location, the reset vector, 0xFFFFFFF0**
- **That means at some point the attacker must hook control flow to gain code execution**
- **That means the attacker changes must be present on the SPI flash chip**
  - Until you start considering additionally compromised peripheral processing units with their DMA capabilities etc
- **If you can get a copy of the flash chip contents, you can integrity check it**
  - You can then either compare all of the same
  - We're working with vendors in the background to make this easier too

MITRE

Integrity Report

- **Integrity check BIOS images against "presumed good" or "known good"**
  - Cannot use simple full-dump hashing for check
    - Some naturally changing portions on UEFI systems
    - Ostensibly the same systems differ on legacy systems (we found we couldn't get an apple-to-apples comparison some times, even from some homogenous systems that arrive in the same shipment!)
  - Decompose UEFI-based firmware filesystem with UEFIExtract, UEFI Firmware Parser, or EFIPWN, and perform pairwise-file comparison
  - Decompose legacy BIOS with scripts that exist on BIOS modders forums :) (they get together to understand BIOS to pirate MS)
  - Parse UEFI non-volatile variables and compare before and after states
    - Some variables naturally change (e.g. monatonic counters) and need to be excluded

MITRE

Integrity Report 2

- **Where do I get a "known good" BIOS?**
  - Can get it from the vendor's website, assuming their network isn't compromised, or your network isn't Man in the Middled
    - If your network is MitMed, get a copy from home too :)
      - If your home is MitMed, get a copy from the library too :P
  - Going to start working with OEMs on a new "BIOS Analysis Metadata Format" (BAMF) which will contain the type of information someone would want to do automated firmware integrity checking

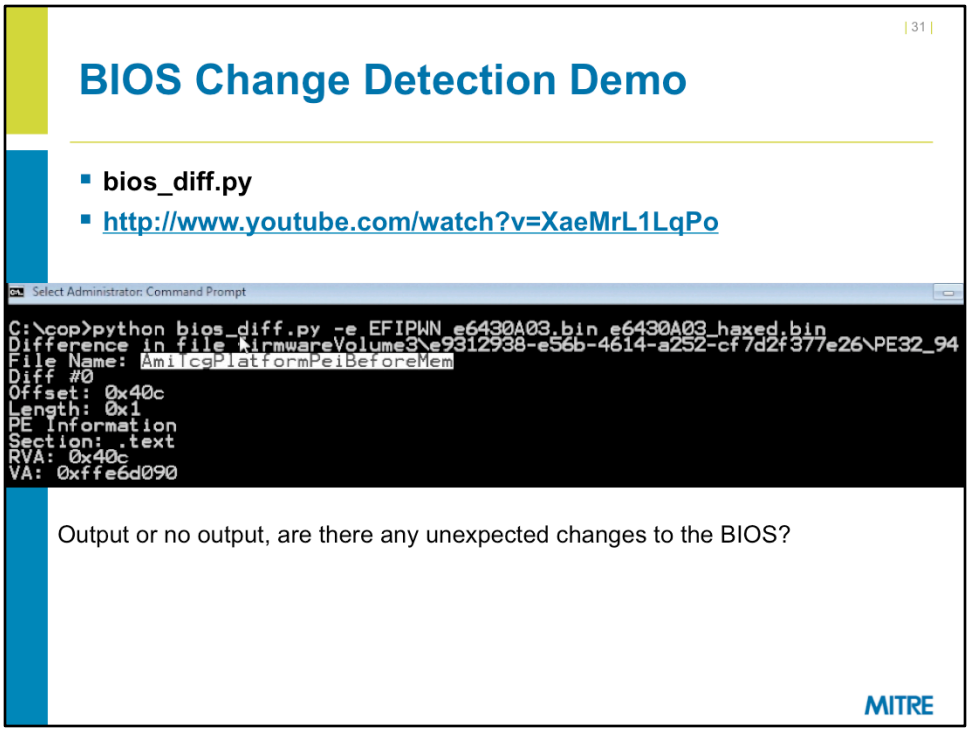
    - Want to pave the way for vendors to start building COTS capabilities. We're basically doing this for *you* VB audience
    - Also will help somewhat with supply chain attacks

MITRE

Integrity Report 3

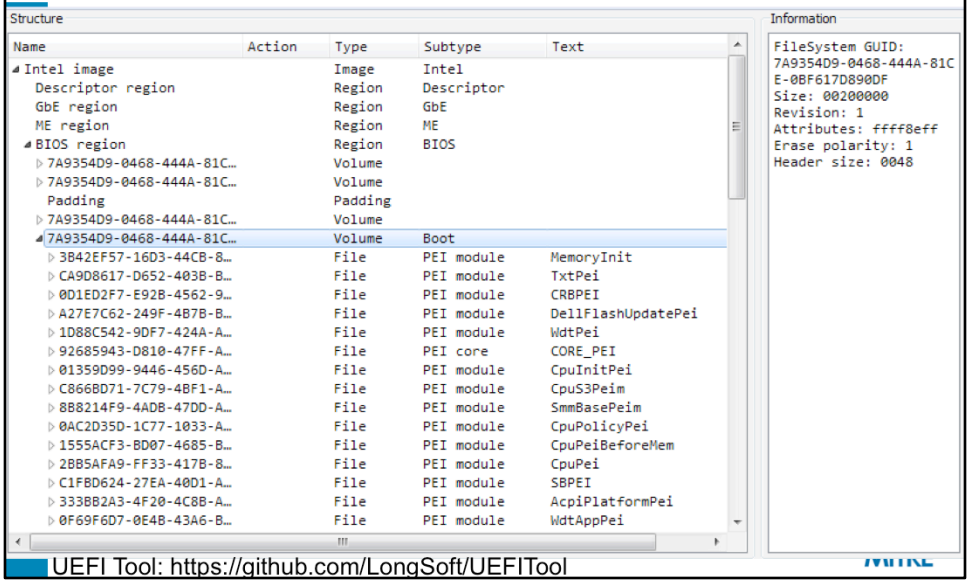
- **Do I need a "known good" BIOS?**
  - No
- **We already have private scripts that dig through a big bucket of BIOSes, sorts them by vendor/model/type, picks a BIOS, calls it "presumed good", and then compares everyone else against that**
  - Assuming #badBIOSes are the minority in the set (e.g. no supply chain attack), then if we picked right, there should be just a couple of outliers which are different, and we focus on those
  - If we picked "wrong", and everything differs from the "presumed good", we would still probably just start by focusing on the outliers
- **Analysis of outliers is tractable as the video in a couple slides shows**

MITRE



Compare two of the same machines over time, and show correlation to the UEFI flash filesystem.
Show one example which is "natural" changes between runs/reboots

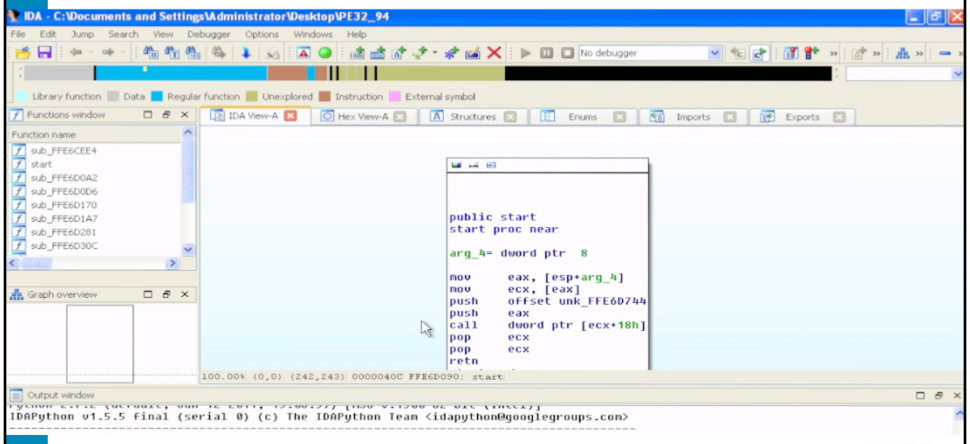
Demo: Using UEFI Tool to Parse FFS



UEFI Tool: https://github.com/LongSoft/UEFITool

Demo: Making sense of UEFI PE files in IDA Pro

- I'm sure I won't have time for this, so peruse at your leisure
- https://www.youtube.com/watch?v=R-5UO6jLkEI



MITRE

Copernicus today

- **Copernicus as a standalone tool has been run on > 10k Windows 7 endpoints. (All of MITRE's + some other organizations')**
- **Available as a free binary-only download**
  - http://www.mitre.org/capabilities/cybersecurity/overview/cybersecurity-blog/copernicus-question-your-assumptions-about or just google "MITRE Copernicus"
- **We want to write a paper on the state of BIOS security in the wild when we get to 100k hosts worth of data**
  - If you'd like to participate in that research, contact me afterwards

MITRE

Copernicus tomorrow

- **Checking the BIOS is no longer research, it needs to be put into practice**
  - Research is now checking the firmware for peripherals like the embedded controllers, HDs, NICs, GPUs, etc
- **MITRE doesn't make products or do long term software support**
  - We're an *R&D center*, it says so right in our FF*RDC* designation!
- **We'd like to get our BIOS techniques incorporated into as many 3rd party security products as possible**
- **We are offering access to the source code source code in exchange for some data being contributed back to us (where architectures allow for it) to help support our ongoing research.**
- **It's a win for all parties; vendors, researchers, and customers**

MITRE

Conclusion

- **Here are the 5 bullet points they asked about for what I want the audience to take away:**
  - We now know that state-sponsored BIOS attackers from multiple countries have existed for years
  - The AV industry hasn't caught them, because they don't inspect at that level
  - UEFI SecureBoot pushes attackers both higher (userspace) and lower (BIOS) in the system
  - The last year has seen a lot of BIOS vulnerabilities revealed (if you for some reason think they need physical access, they don't)
  - The technology to inspect the BIOS for vulnerabilities and integrity attacks is freely available for incorporation into security products, and therefore security vendors should start investigating it
- ***Leaders* in the security industry will take advantage of this opportunity, contact us, and start checking firmware. *Followers* will continue to leave their customers vulnerable, and probably never implement anything. *Wankers* will say "not invented here" and go reinvent the wheel instead of starting from the bicycle they're offered.**

MITRE

Questions?

- Thanks for listening!

- Email contact:

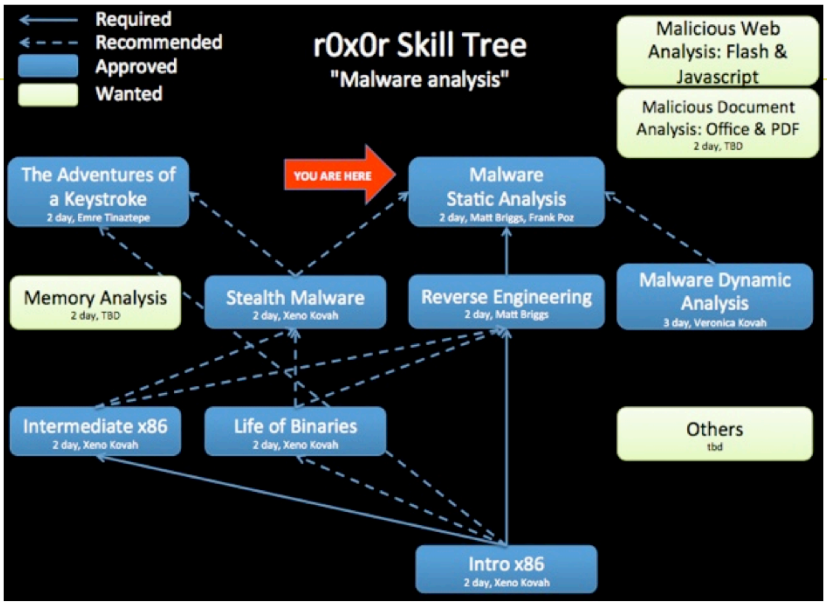
{xkovah, ckallenberg, jbutterworth, scornwell, rheinemann} at mitre dot org

- Twitter contact:

@xenokovah, @coreykal, @jwbutterworth3, @ssc0rnwell

Obligatory "go check out OpenSecurityTraining.info" shout out. Our RE curriculum is the most mature, and it would be a good way for this group to bootstrap new employees

MITRE



MITRE

Copernicus Caveats

- **Only Intel CPU/chipset support, no AMD support**
  - We'll add AMD when someone says they have a lot of those machines and they're willing to contribute data back to us
  - Or when AMD gives us free machines ;)
- **Only supports Windows 7 32 & 64 bit and newer**
  - Doesn't *officially* support Windows 8 but it's been known to run on it for some people, and not for others
  - Adding support for XP and greater, but mainly because we want Win2k3 support and they share a kernel.
- **Bios_diff.py doesn't diff UEFI variables yet**
  - It's on our todo list
- **Fundamentally untrustworthy…a kernel mode attacker can make it lie…just like every other piece of security software you currently use**
  - Copernicus 2 is *much* more trustworthy, but it requires a TPM (with the requisite secure provisioning), and CPU support for Intel TXT, but also doesn't support 64 bit yet because Flicker doesn't support 64 bit
    - Copernicus 3 will be even better :)

MITRE

References

- [1] Attacking Intel BIOS – Alexander Tereshkin & Rafal Wojtczuk – Jul. 2009
  http://invisiblethingslab.com/resources/bh09usa/Attacking%20Intel%20BIOS.pdf
- [2] TPM PC Client Specification - Feb. 2013
  http://www.trustedcomputinggroup.org/developers/pc_client/specifications/
- [3] Evil Maid Just Got Angrier: Why Full-Disk Encryption With TPM is Insecure on Many Systems – Yuriy Bulygin – Mar. 2013
  http://cansecwest.com/slides/2013/Evil%20Maid%20Just%20Got%20Angrier.pdf
- [4] A Tale of One Software Bypass of Windows 8 Secure Boot – Yuriy Bulygin – Jul. 2013
  http://blackhat.com/us-13/briefings.html#Bulygin
- [5] Attacking Intel Trusted Execution Technology - Rafal Wojtczuk and Joanna Rutkowska – Feb. 2009
  http://invisiblethingslab.com/resources/bh09dc/Attacking%20Intel%20TXT%20-%20paper.pdf
- [6] Another Way to Circumvent Intel® Trusted Execution Technology - Rafal Wojtczuk, Joanna Rutkowska, and Alexander Tereshkin – Dec. 2009
  http://invisiblethingslab.com/resources/misc09/Another%20TXT%20Attack.pdf
- [7] Exploring new lands on Intel CPUs (SINIT code execution hijacking) - Rafal Wojtczuk and Joanna Rutkowska – Dec. 2011
  http://www.invisiblethingslab.com/resources/2011/Attacking_Intel_TXT_via_SINIT_hijacking.pdf
- [7] Meet 'Rakshasa,' The Malware Infection Designed To Be Undetectable And Incurable - http://www.forbes.com/sites/andygreenberg/2012/07/26/meet-rakshasa-the-malware-infection-designed-to-be-undetectable-and-incurable/

MITRE

References 2

- [8] Implementing and Detecting an ACPI BIOS Rootkit – Heasman, Feb. 2006 http://www.blackhat.com/presentations/bh-europe-06/bh-eu-06-Heasman.pdf
- [9] Implementing and Detecting a PCI Rookit – Heasman, Feb. 2007 http://www.blackhat.com/presentations/bh-dc-07/Heasman/Paper/bh-dc-07-Heasman-WP.pdf
- [10] Using CPU System Management Mode to Circumvent Operating System Security Functions - Duflot et al., Mar. 2006 http://www.ssi.gouv.fr/archive/fr/sciences/fichiers/lti/cansecwest2006-duflot-paper.pdf
- [11] Getting into the SMRAM:SMM Reloaded – Duflot et. Al, Mar. 2009 http://cansecwest.com/csw09/csw09-duflot.pdf
- [12] Attacking SMM Memory via Intel® CPU Cache Poisoning – Wojtczuk & Rutkowska, Mar. 2009 http://invisiblethingslab.com/resources/misc09/smm_cache_fun.pdf
- [13] Defeating Signed BIOS Enforcement – Kallenberg et al., Sept. 2013
- http://www.syscan.org/index.php/download/get/6e597f6067493dd581eed737146f3afb/SyScan2014_CoreyKallenberg_SetupforFailureDefeatingSecureBoot.zip
- [14] Mebromi: The first BIOS rootkit in the wild – Giuliani, Sept. 2011 http://www.webroot.com/blog/2011/09/13/mebromi-the-first-bios-rootkit-in-the-wild/

MITRE

References 3

- [15] Persistent BIOS Infection – Sacco & Ortega, Mar. 2009 http://cansecwest.com/csw09/csw09-sacco-ortega.pdf
- [16] Deactivate the Rootkit – Ortega & Sacco, Jul. 2009 http://www.blackhat.com/presentations/bh-usa-09/ORTEGA/BHUSA09-Ortega-DeactivateRootkit-PAPER.pdf
- [17] Sticky Fingers & KBC Custom Shop – Gazet, Jun. 2011 http://esec-lab.sogeti.com/dotclear/public/publications/11-recon-stickyfingers_slides.pdf
- [18] BIOS Chronomancy: Fixing the Core Root of Trust for Measurement – Butterworth et al., May 2013 http://www.nosuchcon.org/talks/D2_01_Butterworth_BIOS_Chronomancy.pdf
- [19] New Results for Timing-based Attestation – Kovah et al., May 2012 http://www.ieee-security.org/TC/SP2012/papers/4681a239.pdf

MITRE

References 4

- [20] Low Down and Dirty: Anti-forensic Rootkits - Darren Bilby, Oct. 2006
  http://www.blackhat.com/presentations/bh-jp-06/BH-JP-06-Bilby-up.pdf
- [21] Implementation and Implications of a Stealth Hard-Drive Backdoor – Zaddach et al., Dec. 2013
  https://www.ibr.cs.tu-bs.de/users/kurmus/papers/acsac13.pdf
- [22] Hard Disk Hacking – Sprite, Jul. 2013
  http://spritesmods.com/?art=hddhack
- [23] Embedded Devices Security and Firmware Reverse Engineering - Zaddach & Costin, Jul. 2013
  https://media.blackhat.com/us-13/US-13-Zaddach-Workshop-on-Embedded-Devices-Security-and-Firmware-Reverse-Engineering-WP.pdf
- [24] Can You Still Trust Your Network Card – Duflot et al., Mar. 2010
  http://www.ssi.gouv.fr/IMG/pdf/csw-trustnetworkcard.pdf
- [25] Project Maux Mk.II, Arrigo Triulzi, Mar. 2008
  http://www.alchemistowl.org/arrigo/Papers/Arrigo-Triulzi-PACSEC08-Project-Maux-II.pdf

MITRE

References 5

- [26] Copernicus: Question your assumptions about BIOS Security – Butterworth, July 2013 http://www.mitre.org/capabilities/cybersecurity/overview/cybersecurity-blog/copernicus-question-your-assumptions-about
- [27] Copernicus 2: SENTER the Dragon – Kovah et al., Mar 2014 https://cansecwest.com/slides/2014/Copernicus2-SENTER_the-Dragon-CSW.pptx
- [28] Playing Hide and Seek with BIOS Implants – Kovah, Mar 2014 http://www.mitre.org/capabilities/cybersecurity/overview/cybersecurity-blog/playing-hide-and-seek-with-bios-implants
- [29] Setup for Failure: Defeating UEFI – Kallenberg et al., Apr 2014 http://syscan.org/index.php/download/get/6e597f6067493dd581eed737146f3afb/SyScan2014_CoreyKallenberg_SetupforFailureDefeatingSecureBoot.zip
- [30] SENTER Sandman: Using Intel TXT to Attack BIOSes – Kovah et al., June 2014 slides not posted anywhere yet
- [31] Extreme Privilege Escalation on UEFI Windows 8 Systems – Kallenberg et al., Aug 2014 - https://www.blackhat.com/docs/us-14/materials/us-14-Kallenberg-Extreme-Privilege-Escalation-On-Windows8-UEFI-Systems.pdf

MITRE