Common BMC Vulnerabilities
And How to Avoid Repeating Them

Rick Altherr
Principal Software Engineer
rick.altherr@eclypsium.com
So What is a BMC?

- Baseboard Management Controller
- Independent subsystem for platform management
- Hosts Intelligent Platform Management Interface (IPMI) and DMTF Redfish stacks
  - Monitoring health of system hardware
  - System power and reset control
  - Logging and alerting of abnormal system conditions
  - Inventory of system components
- Often provides “remote hands” capabilities
  - Virtual console (aka iKVM)
  - Virtual media (mount ISO as virtual optical drive on host)
  - Host firmware update
Terminology

- **Host**
  Computer intended to run end-user applications

- **Server**
  Device containing both a host and a BMC

- **Chassis**
  Physical enclosure containing at least one server. May contain its own BMC independent of server BMCs.
## Common BMC SoCs

<table>
<thead>
<tr>
<th>SoC Family</th>
<th>CPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Engines/Emulex/Aspeed Pilot4</td>
<td>ARM Cortex A9</td>
</tr>
<tr>
<td>Aspeed AST2400</td>
<td>ARM 926EJ</td>
</tr>
<tr>
<td></td>
<td>CoreFire V1</td>
</tr>
<tr>
<td>Aspeed AST2500</td>
<td>ARM 1176JZS</td>
</tr>
<tr>
<td></td>
<td>32-bit RISC (ColdFire?)</td>
</tr>
<tr>
<td>HP iLO4</td>
<td>ARM</td>
</tr>
<tr>
<td>HP iLO5</td>
<td>ARM</td>
</tr>
<tr>
<td>Nuvoton WPCM450</td>
<td>ARM 926EJ-S</td>
</tr>
<tr>
<td>Nuvoton NPCM750</td>
<td>ARM Cortex A9</td>
</tr>
<tr>
<td></td>
<td>CompactRisc CR16C+</td>
</tr>
<tr>
<td>Renesas SH7758</td>
<td>SuperH SH-4A</td>
</tr>
</tbody>
</table>
Why Attack a BMC?

- Highly privileged access to host
- Often network accessible
- Persistence independent from host
- Poor firmware security history
- Firmware updates infrequent and rarely applied
Prior Research


  [https://blog.rapid7.com/2013/07/02/a-penetration-testers-guide-to-ipmi/](https://blog.rapid7.com/2013/07/02/a-penetration-testers-guide-to-ipmi/)

  [https://www.synacktiv.com/ressources/recon_bx_2018_ilo4_perigaud_gazet_czarny.pdf](https://www.synacktiv.com/ressources/recon_bx_2018_ilo4_perigaud_gazet_czarny.pdf)
Common Vulnerability #1
Implementing IPMI to the spec
RMCP+ Weak Crypto

- Every mandatory crypto algorithm is broken or weak
- Modern algorithms aren't even available as options
- IPMI spec final update in 2013
- Even IPMI Promoters recommend using something else.²

Password Storage

- Required session authentication algorithms
  - Plaintext password
  - MD2
  - MD5

- RAKP authentication uses HMAC insecurely
  - Server sends salted hash of user password before client authenticates\(^1\)
  - Metasploit Framework has script to collect hash this way
  - John the Ripper and hashcat can crack them
  - Cost to crack all 1-8 character alphanumeric passwords => ~$20

- For both of these cases, user passwords must be stored in plaintext in the BMC

\(^1\)http://fish2.com/ipmi/remote-pw-cracking.html
Unauthenticated Interfaces

- Local interfaces provide session-less (and thus unauthenticated) channels
- Nearly any command is allowed via these channels on IPMI v1.5 and earlier
- IPMI v2.0 introduced Firmware Firewall
  - New commands to disable individual commands
  - Supports per-channel configuration
  - Optional part of specification
  - If present, no recommended default configuration
  - In practice, many vendors default to IPMI v1.5 behavior
- **TL;DR** Host and add-in cards have unrestricted, unauthenticated IPMI access
Common Vulnerability #2
Forgetting About Web Application Security
Typical BMC Web App Stack

1. Unvalidated Input
2. Broken Access Control
3. Broken Authentication and Session Management
4. Cross-Site Scripting Flaws
5. Buffer Overflows
6. Injection Flaws
7. Improper Error Handling
8. Insecure Storage
9. Denial of Service
10. Insecure Configuration Management
OWASP Top 10 (2004) as Found in BMCs

1. Unvalidated Input
2. Broken Access Control
3. Broken Authentication and Session Management
4. Cross-Site Scripting Flaws
5. Buffer Overflows
6. Injection Flaws
7. Improper Error Handling
8. Insecure Storage
9. Denial of Service
10. Insecure Configuration Management

CVE-2018-1211 (Dell iDRAC7/8)
CVE-2013-3609 (Supermicro)
CVE-2019-3706 (Dell iDRAC9)
CVE-2015-7275 (Dell iDRAC6/7/8)
CVE-2019-3705 (Dell iDRAC7/8/9)
CVE-2018-1207 (Dell iDRAC7/8)
CVE-2018-7101 (HP iLO4)
CVE-2017-12542

- Authentication bypass and remote code execution in HP iLO4 v2.52 and earlier
- HTTP request parsed line by line with strcmp(), strstr(), and sscanf()
- “Connection” header value copied into 16-byte buffer
- Exploit:
  
Lack of Privilege Separation

Why are these running as root?!?!
Common Vulnerability #3
Unsigned Firmware and Updates
What’s in a Supermicro X10 BMC update?

- Typical embedded Linux flash layout
- JFFS2 partition blank in firmware updates
- `[img]` section

```
strings -a REDFISH_X10_327.bin | fgrep '[img]' |
| sed -e 's/\[/\n\[/g'
```

```
[img]: 0 20fec cf74e74e u-boot.bin
[img]: 400000 d28000 ec25e35d out_rootfs_img.bin
[img]: 1400000 177620 64d09306 out_kernel.bin
[img]: 1700000 55f00a ed586d8c out_webfs_img.bin
[end]
```

<table>
<thead>
<tr>
<th>Start offset</th>
<th>End offset</th>
<th>CRC32</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>128k</td>
<td>U-Boot Bootloader</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JFFS2 for <code>/nv</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CramFS for <code>/</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kernel</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CramFS for <code>/web</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>[img]</code> tags</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>padding</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2019 Eclypsium
What’s in a Supermicro X11 BMC update?

- Identical to X10
- Hardcoded AES keys and IVs are used to encrypt [img] tags and first 96 bytes of CramFS regions
- Unencrypted when flashed to real system
- Easy to extract encryption keys

```c
j_console_log("[\%s] img-size: \%#x\n", "flash_decrypt_check", a2);
j_console_log("[\%s] Flash encryption check ...
", "flash_decrypt_check");
v3 = j_crypto_task1(v2, 2, 0x400000);
if ( v3 >= 0 )
{
v3 = j_crypto_task1(v2, 2, 0x1700000);
if ( v3 >= 0 )
{
v5 = j_crypto_task2(v2, 2);
```
What about secure boot?

- Only the newest BMC SoCs have support
  - Nuvoton NPCM750
  - HP iLO5
  - Aspeed AST2600 (so I’ve been told)
- Requires full chain of verification
  - SoC -> bootloader
  - Bootloader -> kernel
  - Kernel -> filesystems
- What do you do when verification fails?
Common Vulnerability #4
Assuming a Trusted Management Network
Typical BMC Deployment
Advanced Lateral Movement

Internet → Private cloud
Advanced Lateral Movement
Advanced Lateral Movement
Advanced Lateral Movement
Advanced Lateral Movement
Advanced Lateral Movement
BMCs on the Internet

Top Countries

1. United States  29,835
2. Russian Federation  8,050
3. Netherlands  6,469
4. Germany  6,340
5. China  4,892
6. United Kingdom  4,531
7. Canada  2,942
8. Korea, Republic of  2,728
9. Turkey  1,944
10. Ukraine  1,523

Search for ports25 returned 94,445 results on 19-08-2019
Common Vulnerability #5

Trusting the Host
Why shouldn’t I trust the host?

- Bare-metal VM
  - Host is running tenant code, possibly hostile
  - BMC is operator’s sole way to gather telemetry and regain control

- Attacker already compromised a host
  - Wants host-independent persistence
  - Looking for lateral movement

- Company policies
  - IT manages hardware (aka BMC)
  - Business unit manages software (aka host)
  - Parallels bare-metal VM model
NC-SI Traffic Intercept

- NC-SI allows BMC and host to share a single NIC
  - Host receives all traffic by default
  - BMC requests specific traffic by configuring filters in NIC
- **Project Ortega** reverse engineering Broadcom BCM5719
  - Goal is clean-room, opensource reimplementation of firmware
  - Intermediate result is documentation of chip internals and firmware behavior
- APE (microcontroller inside NIC) implements all NC-SI handling
  - Cortex-M3 with compressed firmware loaded from PCI Option ROM
  - Firmware is unsigned
- Host can load malicious firmware that intercepts all BMC network traffic
MCTP over PCIe Intercept

- Management Component Transport Protocol (MCTP) forms an overlay network for management traffic on top of various common buses already present in systems
- MCTP over PCIe uses Vendor-Defined Messages (VDMs) as specified in PCIe Base Specification
- VDMs can be routed: to root complex, broadcast from root complex, or to device by ID
- Device by ID routing is effectively peer-to-peer PCIe traffic
  - Which a root complex is not required to support or allow
  - When disabled, the root complex acts as a Man-in-the-Middle
MCTP over PCIe With Peer-to-Peer

- Root allows peer-to-peer traffic
- BMC sends MCTP VDM to NIC using Device by ID routing
- MCTP VDM travels upstream to PCIe switch
- PCIe switch sees destination is on a downstream port and forwards VDM
- NIC receives VDM
MCTP over PCIe Without Peer-to-Peer

- Root **disallows** peer-to-peer traffic
- BMC sends MCTP VDM to NIC using Device by ID routing
- MCTP VDM travels upstream to PCIe switch
- PCIe switch sees destination is on a downstream port but P2P is disabled so it sends to upstream
- PCIe Root receives VDM and resends downstream toward destination
- PCIe switch forwards to downstream port
- NIC receives VDM
Threat Model
Every Interface Is Hostile
Assumptions

- Case-open, physical attacks are out of scope
- Bare-metal VM use case
  - Operator uses BMC for telemetry and to regain control
  - Tenant has full use of host and may be hostile
- BMC is highest-value device in server
  - May be Root of Trust
  - Has complete control over host
Typical BMC SoC Architecture Diagram
Typical BMC SoC Architecture Diagram
Host <-> BMC

- LPC/eSPI
  - What I/O ports can the host access?
  - What BMC software acts on those requests?
- PCIe
  - What address mappings share resources between the host and BMC?
  - Are those sufficiently restricted in BMC address space?
  - Is BMC software reading from those shared resources treating the data as hostile?
- USB Virtual Hub
  - What data am I trusting from the host?
  - How can the host cause my virtual device to misbehave?
LAN <-> BMC

- What services are accessible?
- How do I know if an individual request is authorized?
- What can an unauthorized user learn and/or cause to happen?
- Have I limited the capabilities of services to only cover their intended usage?
- Do I need to limit capabilities per-user or per-role?
- Does every service use strong encryption for every request?
External Connectors <-> BMC

- USB Host
  - What happens if an unexpected device is attached?
  - If you expect mass storage devices, how do you verify their contents?
- VGA (DE-15)
  - Where is the framebuffer located?
  - What data will be shown?
- RS-232 (DE-9)
  - What data will be transferred?
Virtual Media Vulnerability in BMC Opens Servers to Remote Attack
Responsible Disclosure Timeline

- 2019-06-19 - Eclypsium reports vulnerability to Supermicro
- 2019-07-09 - Eclypsium reports additional findings to Supermicro
- 2019-07-29 - Supermicro acknowledges report and develops a fix
- 2019-08-16 - Eclypsium notifies CERT/CC due to large number of public systems affected
- 2019-08-16 - Supermicro confirms intent to publicly release firmware by September 3rd
- 2019-08-20 - Eclypsium dubs the vulnerabilities USBAnywhere
- 2019-08-23 - Eclypsium begins notifying network operators whose networks contain affected, Internet-accessible BMCs
- 2019-08-23 - Eclypsium discovers that Supermicro X9 platforms are also affected
- 2019-09-03 - Eclypsium publishes vulnerability details and presents same at Open Source Firmware Conference
How Does Virtual Media Work?
What We Know

- Java applet launched via JNLP
- ISO located on system running Java applet
- “Plugging in” the ISO attaches a USB device on the remote host
  - **USB Class**: Mass Storage
  - **USB Subclass**: SCSI Transparent Command Set
  - **SCSI PDT**\(^1\): Multimedia Commands (MMC)
  - ATEN Virtual CDROM
- iKVM also uses USB for virtual keyboard and mouse

\(^1\)SCSI Peripheral Device Type
Connections Between Host and BMC
Virtual USB Hub

Hub Device

Downstream Device

Downstream Device

Downstream Device

Downstream Device

Downstream Device

Endpoint

Endpoint

Endpoint

Endpoint

Endpoint

Endpoint

Endpoint

Endpoint

Endpoint

Endpoint

Endpoint
What’s Going Over the Network?

- HTTP
  - JNLP launcher
  - Java JARs
- VNC
  - iKVM
- TCP/623
  - Started when Virtual Media UI opened
Unencrypted USB over TCP?!?!

SCSI Vendor and Product IDs

USB Mass Storage Class (MSC)
Bulk-only Transport (BOT)
Command Status Wrapper (CSW)
Signature
Understanding the Protocol

Frame 10597: 280 bytes on wire (2240 bits), 280 bytes captured (2240 bits) on Interface 0
Ethernet II, Src: 00:ff:bf:78:99:82 (00:ff:bf:78:99:82), Dest: 00:ff:c0:78:99:82 (00:ff:c0:78:99:82)
Internet Protocol Version 4, Src: 10.0.8.5, Dest: 172.16.0.11
Supermicro Virtual Media, Tag: Device Setup, Port 0, Endpoint: 0, Len: 226 (encrypted)
Tag: Device Setup (0x01)
  Device Port: 0
  Endpoint: 0x00
    0000 = Number: 0
    0000 ... = Type: Unknown (0)
  Flags: 0x00
    1...... = Encrypted: True
  Payload Length: 44
Virtual Media Device setup Request
  Username: t9v4pfb90060M
  Password: eXm9bIa=-
  Flags: 0x8051
    1.... = Username IS session ID: True
    0... = Check auth only: False
    ....01 = Requested Port: 1
    ......1 = Allocate Port: True

Frame (280 bytes)  Decrypted Payload (218 bytes)

© 2019 Eclypsium
Understanding the Protocol

Encryption is optional

So are plaintext username/password
It Gets Worse

- Encryption
  - RC4 with same fixed key on all X9, X10, and X11 systems
  - Could have been used for every packet but wasn’t

- USB device is implemented entirely client-side
  - Server caches client-provided USB descriptors
  - Almost all endpoint traffic sent directly to client

- Authentication bypass on X10 and X11
  - Credentials cached by socket file descriptor
  - Client disconnect fails to invalidate cache
  - Very high chance of unintentional reuse by a new client
Making My Own Client

- Facedancer\(^1\) is a Python framework for emulating USB devices
  - Originally designed for use with special-purpose hardware
  - Now has a plugin architecture for backends
- USBAnywhere backend
  -_opts to not use encryption
  - Uses plaintext username/password auth
  - PoC quality

\(^1\)https://github.com/usb-tools/Facedancer
Impact

- 47,000 affected BMCs found on the Internet
  - 1905 Autonomous Systems (AS)
  - 90+ countries
- How many are on your enterprise network?
- Attack scenarios
  - Exfiltrate data over virtual USB mass storage device
  - Boot machine from attacker-provided ISO
  - Network-attached USB Rubber Ducky\(^1\)
  - and the list goes on...

\(^1\)https://shop.hak5.org/products/usb-rubber-ducky-deluxe
Putting It All Together
For End Users

- Protect your existing BMCs
  - Existing BMCs are very privileged and very vulnerable
  - Treat them like an unpatched Windows XP host on your network

- Update your infrastructure
  - Adopt Redfish as a replacement for IPMI
  - Fix anything preventing you from using a hardened BMC

- Put pressure on your vendors to do better
  - Include BMC security as part of your selection criteria
  - Ask about BMC roadmap
For Original Equipment Manufacturers

- Establish a Security Response Team (SRT)
  - Single point of contact for reporting security issues across all products
  - Draw from ISO 30111\textsuperscript{1}: Vulnerability Handling Process

- Make it easy to deploy BMCs safely
  - Design defaults assuming BMC will be on the Internet
  - Generate per-device default passwords and certificates
  - Guide customers toward modern protocols (Redfish, HTTPS, etc)

- Challenge BMC firmware and SoC vendors to do better
  - Require secure boot
  - Ask for security audit results

\textsuperscript{1}https://www.iso.org/standard/53231.html
For Developers

- Design defensively
  - Expect every feature to be misused
  - Regularly ask yourself “What can go wrong?”

- Insist on improving security
  - Design-in secure boot as mandatory
  - Prioritize implementing modern protocols

- Test security in your CI process
  - Find the breakage as fast as possible
  - Block release on security test failures
USBAnywhere Resources

- Eclypsium Blog
- Proof-of-concept Demo Video
  https://youtu.be/8UI7oicMisY
- Tools, Packet Captures, etc
  https://github.com/eclypsium/USBAnywhere
## BMC/IPMI History

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IPMI v1.0 spec</strong></td>
<td><strong>IPMI v1.5 spec</strong></td>
<td><strong>IPMI v2.0 spec</strong></td>
<td><strong>Many BMC/IPMI vulnerabilities published</strong></td>
<td><strong>SMC PSBlock password file vulnerability</strong></td>
<td><strong>HP iLO4 auth bypass and RCE</strong></td>
</tr>
<tr>
<td>Base version of IPMI specification released</td>
<td>Many enhancements to base specification including IPMI over LAN and IPMI over Serial/Modem</td>
<td>New features including Serial over LAN, Enhanced Authentication, Firmware Firewall, and VLAN support</td>
<td>Dan Farmer and HD Moore found over 300k BMCs connected to the internet, 53k vulnerable to cipher-zero auth bypass</td>
<td>Zachary Wikholm discovered that Supermicro BMCs have plaintext password file which could be retrieved remotely without auth, 32k on internet</td>
<td>Multiple vulns including trivial auth bypass: <code>curl -H &quot;Connection: AAAAAAAAAAAA AAAAAAAAAAAA AAA&quot;</code></td>
</tr>
</tbody>
</table>
USB Internals

- USB Device comprised of endpoints
- Endpoints are unidirectional sources or sinks
- Direction is relative to host
  - OUT = host out to device
  - IN = host in from device
- Endpoint 0 is required for control/status
- Other endpoints and how they are used are discovered via descriptors
USB Descriptors

- Hierarchy of descriptive information
  - Identification
  - USB compatibility
  - Device class/subclass
- Configuration
  Mutually exclusive way of using the device
- Interface
  Non-exclusive logical function provided by the device
Understanding the Protocol

00 80 00 01 2c 00 00 00 ...

- Payload Length (LE32)
- Tag
- Device Port
- Flags
- Endpoint