IPMI:
Understanding Your Server’s Remote Backdoor

Anthony J. Bonkoski
abonkosk@umich.edu

SUMIT 2013
What is IPMI?

Need to manage a massive cluster of servers?
- Operating system installs
- Monitoring
- Power cycling
- Remote Virtual Desktop

Impractical to physically administer each machine
What is IPMI?

Intelligent Platform Management Interface (IPMI) Specification by Intel:

- Adds a second computer
- Always on
- Acts as a gateway to the machine
- Integrated directly into the system buses (e.g. I²C)
What is IPMI?

Several Brand Names:

HP iLo
Dell iDrac
Sun/Oracle iLOM
Lenovo/IBM IMM
SuperMicro IPMI
ATEN IPMI
MegaRAC
Avocent IPMI
What is IPMI?

Second Computer: Baseboard Management Controller (BMC)
Typical IPMI Implementation

System

Embedded on Motherboard or Expansion card
CPU: ARM/MIPS or other low power embedded CPU
OS: Linux or a lightweight real-time OS

Extra OEM Features

Remote Virtual Console
Remote Media
High network connectivity incl. HTTP and SSH.
Motivation

IPMI is the perfect spying backdoor
Always on and often pre-enabled.
NIC failover*
Powerful Remote Tools
Widespread deployment:
  100,000+ on public IPs
  millions on private networks

It’s an embedded system...
...often, security is an after-thought!

*As seen on our SuperMicro ATEN-based IPMI
Motivation

But IPMI is not new:

- IPMI 1.0: 9/16/1998
- IPMI 1.5: 3/1/2001
- IPMI 2.0: 2/14/2004

Why do we care now?

- Most servers ship with IPMI and often pre-enabled
- Many vendors add “special” features in addition to the base specification:
  - SSH, Remote KVM, Remote Virtual Media, HTTP, etc.
- Today’s IPMI is “not your grandfather’s”

*As seen on our SuperMicro ATEN-based IPMI*
Dan Farmer

January 2013: Starts publicly denouncing IPMI

“IPMI: Freight Train to Hell”: lots of conjectures

Hidden backdoor debugging web page on Dell iDRAC

Could gain root over ssh
Recent Developments

Dan Farmer

IPMI Specification is broken

1) Cipher-0 allows a user to bypass crypto, and login to BMC locally as admin
2) IPMI 2.0 RAKP Password Hash Retrieval
   Attacker can retrieve any user’s password hash and crack it offline
3) Anonymous authentication
4) Cleartext passwords for calculating authentication hash
Our Work

Lets look inside the black-box:

- We reverse-engineered a popular OEM’s firmware
- We examined the web interface implementation

Paper

Anthony Bonkoski, Russ Bielawski, and J. Alex Halderman
Published in Usenix Workshop on Offensive Technologies (WOOT) 2013
Supermicro IPMI

Supermicro SYS-5017C-LF

IPMI Firmware by ATEN Technology

- HTML / JavaScript
- CGI (written in C)
- Linux 2.6.17
- Firmware version 1.86 (build date: 11-14-2012)

Nuvoton WPCM450
ARM-based BMC
Supermicro Web Interface

Summary
Firmware Revision: 01.86
Firmware Build Time: 2012-11-14

IP address: [redacted]
BMC MAC address: 00:25:90:ac:1b:88
System LAN1 MAC address: 00:25:90:ac:7:34
System LAN2 MAC address: 00:25:90:ac:7:35

Remote Console Preview

Power Control via IPMI
Host is currently on

Power On  Power Down  Reset
Supermicro SSH Interface

Backend: Highly modified fork of Dropbear

Frontend: Systems Management Architecture for Server Hardware Command-Line Protocol (SMASH)*

Notice: a system admin has no access to underlying Unix shell

*Distributed Management Task Force (DMTF) specification: dmtf.org/standards/smash
Reverse Engineering Approach

Fetch firmware from OEM website.

Scan and unpack: binwalk

<table>
<thead>
<tr>
<th>DECIMAL</th>
<th>HEX</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>59700</td>
<td>0xE934</td>
<td>Copyright string:</td>
</tr>
<tr>
<td>60835</td>
<td>0xEDA3</td>
<td>Copyright string:</td>
</tr>
<tr>
<td>1572864</td>
<td>0x180000</td>
<td>CramFS filesystem,</td>
</tr>
<tr>
<td>9961472</td>
<td>0x980000</td>
<td>Zip archive data,</td>
</tr>
<tr>
<td>11086483</td>
<td>0xA92A93</td>
<td>End of Zip archive</td>
</tr>
<tr>
<td>12058624</td>
<td>0xB80000</td>
<td>CramFS filesystem,</td>
</tr>
</tbody>
</table>

Mount filesystems
Reverse Engineering Approach

Exploring Code:
1) Client-side Javascript
2) Disassembling server-side

Looking for the Classics:
1) Insecure Input Validation
2) Shell Injection
3) Buffer Overflows

```
STMIA R8, {R3,R4} ; Store Block to
MOV R2, #4 ; Rd = 0p2
MOV R1, R5 ; Rd = 0p2
MOV R0, R6 ; Rd = 0p2
BL memcpy ; Branch with Lin
LDR R1, =unk_0_10018870 ; Load from M
ADD R0, R9, #0x970 ; Rd = 0p1 + 0p2
ADD R0, R0, #0xC ; Rd = 0p1 + 0p2
BL Copy_5TDes8RC6TDesC8 ; TDes8::Co
LDR R3, =0x10AC ; Load from Mem
MOV R4, #0 ; Rd = 0p2
STR R4, [R9,R3] ; Store to Memory
LDR R3, =0x1080 ; Load from Mem
MUL R2, #0 ; Rd = ~0p2
STR R2, [R9,R3] ; Store to Memory
MOV R1, R4 ; Rd = 0p2
MOV R0, #0x10000 ; Rd = 0p2
BL __un_FUi6TLeave ; operator new[]
STR R0, [R9,#0x42C] ; Store to Memory
MOV R1, R4 ; Rd = 0p2
```
Input Validation

All input validation and permission checking is...
...in client-side javascript

```javascript
function PrivilegeCallBack(Privilege) {
    // full access
    if (Privilege == '04') {
        isSuperUser = 1;
    }
    // only view
    else if (Privilege == '03') {
        var_save_btn.disabled = true;
    }
    // no access
    else {
        alert(lang.LANG_NOPRIVI);
    }
}
```
Input Validation

Server-side?

No permission checking.
No escaping of input passed to shell.
No string length checking in CGI.

Server *only* creates a session token!
Input Validation

We can send any input to the CGI programs...

...even `rm -fr /`

Will it get executed? Let’s look for calls to `system()`.

15 of 67 CGI programs made calls to `system()`.
Confirmed shell injection in config_date_time.cgi:
Shell Injection

Confirmed shell injection in *config_date_time.cgi*: 

```
Primary NTP Server: 127.0.0.1 `sleep 60`
```
Confirmed shell injection in `config_date_time.cgi`:

![Primary NTP Server: 127.0.0.1 `sleep 60`]

Getting command output

Redirect to `/nv/system_log`.

Issue GET request to `system_log.cgi`. 
Shell Injection

Confirmed shell injection in `config_date_time.cgi`:

[Image: Primary NTP Server: 127.0.0.1 `sleep 60`]

Getting command output

Redirect to `/nv/system_log`.
Issue GET request to `system_log.cgi`.

Create a pseudo-terminal

Wrap POST and GET requests in a python script.

`root@ipmi #`
Shell Injection

Does this give an attacker full remote access?

Not quite. This attack requires a user account.
More reverse engineering

// login.cgi
int main(void)
{
    char name[128], pwd[24];
    char *temp ;
    // ... initialize ...
    temp = cgiGetVariable("name");
    strcpy(name, temp);
    temp = cgiGetVariable("pwd");
    strcpy(pwd, temp);
    // ... authenticate user ...
}
More reverse engineering

// login.cgi
int main(void)
{
    char name[128], pwd[24];
    char *temp;
    // ... initialize ...
    temp = cgiGetVariable("name");
    strcpy(name, temp);
    temp = cgiGetVariable("pwd");
    strcpy(pwd, temp);
    // ... authenticate user ...
}

Buffer Overflows!
In the login page!
Buffer Overflows

No length validation?
Buffer Overflows

No length validation?

```html
<input name="name" size="20" maxlength="64"
```
Buffer Overflows

No length validation?

```html
<input name="name" size="20" maxlength="1000"
```
Buffer Overflows

No length validation?

```html
<input name="name" size="20" maxlength="1000"

500 - Internal Server Error
```
Buffer Overflow Exploitability

Buffer-overflow defenses?
Buffer Overflow Exploitability

Buffer-overflow defenses?

No DEP (Stack and Heap are executable).
Buffer Overflow Exploitability

Buffer-overflow defenses?
No DEP (Stack and Heap are executable).
No Stack Canaries.
Buffer Overflow Exploitability

Buffer-overflow defenses?
No DEP (Stack and Heap are executable).
No Stack Canaries.
Limited ASLR.

(Stack/Heap base addresses are randomized, but dynamic libraries are not. Return-to-libc works.)
Exploitation Challenges

Stack memory addresses are randomized (ASLR).

...but, only 12 bits are random. Just 4096 possibilities.
Exploitation Challenges

Stack memory addresses are randomized (ASLR).
...but, only 12 bits are random. Just 4096 possibilities.

We gain control on the return from main().
Stack is small: shellcode must be compact.
Exploitation Challenges

Stack memory addresses are randomized (ASLR).
...but, only 12 bits are random. Just 4096 possibilities.

We gain control on the return from main().
Stack is small: shellcode must be compact.

BMC crashes and reboots if pounded too hard with requests.
Solutions

Store the shell command in the name buffer. Brute force through the stack randomization. Limit the time between brute-force iterations.

Avg. search time: ~7 min.
Buffer Overflow Exploit

Solutions
Store the shell command in the name buffer.
Brute force through the stack randomization.
Limit the time between brute-force iterations.
Avg. search time: \(~7\) min.

Payload
Fetch (wget) and install modified SSH daemon.
Forks root shell on incorrect password.
Only 2 instructions changed!

```
root@ipmi #
```
Vulnerable Models

Cursory check of all Supermicro IPMI firmware downloads as of May 23, 2013.

30 of 64 images appear vulnerable.

135 device models.

Vulnerability disclosed to CERT Vulnerability Note VU#648646.

Supermicro says they have a fix, but we haven’t directly verified.

Possibly affects other ATEN-based products.
The Impact

So, rooting this device is *easy*!
But, what are the implications?

Yet another broken embedded system?
The Impact

Only as *secure* as our weakest component. Entire system is now vulnerable! Adding an entire computer only weakens.
IPMI for Evil

BMC-based spyware and botnets
IPMI for Evil

BMC-based spyware and botnets

Rooted BMC → Rooted host system

Mount a custom OS and reboot.
IPMI for Evil

BMC-based spyware and botnets

Rooted BMC $\rightarrow$ Rooted host system
Mount a custom OS and reboot.

Rooted host system $\rightarrow$ Rooted BMC
Re-flash the BMC with malicious code.
IPMI for Evil

BMC-based spyware and botnets

Rooted BMC $\rightarrow$ Rooted host system
  Mount a custom OS and reboot.

Rooted host system $\rightarrow$ Rooted BMC
  Re-flash the BMC with malicious code.

BMC rootkits
  A backdoor that survives potentially forever.
IPMI for Evil

BMC-based spyware and botnets

Rooted BMC $\rightarrow$ Rooted host system
Mount a custom OS and reboot.

Rooted host system $\rightarrow$ Rooted BMC
Re-flash the BMC with malicious code.

BMC rootkits
A backdoor that survives potentially forever.

A scary thought
IPMI meets Matrix $\rightarrow$ Is your IPMI just emulated? How do you know?
Scanned all public IPs on May 7, 2013 using ZMap.

Downloaded all X.509 certs from HTTPS servers.

Used identifying characteristics of default certificates.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Devices on Public IPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supermicro IPMI</td>
<td>41,545</td>
</tr>
<tr>
<td>Dell iDARC</td>
<td>40,413</td>
</tr>
<tr>
<td>HP iLO</td>
<td>23,376</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>105,334</strong></td>
</tr>
</tbody>
</table>

† Details on “identifying characteristics” may be found in our paper
Scanned all public IPs on May 7, 2013 using ZMap.

Downloaded all X.509 certs from HTTPS servers.

Used identifying characteristics of default certificates.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Devices on Public IPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supermicro IPMI</td>
<td>41,545</td>
</tr>
<tr>
<td>Dell iDARC</td>
<td>40,413</td>
</tr>
<tr>
<td>HP iLO</td>
<td>23,376</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>105,334</strong></td>
</tr>
</tbody>
</table>

† Details on “identifying characteristics” may be found in our paper
Defenses

For System Operators

*Never* attach your IPMI device directly to the Internet.

Use an isolated management network or VLAN.

Change default passwords and certificates.

Disable IPMI if you don’t need it.

Unfortunately: we’re at the will of the Vendor
Lessons

A Culture Clash?

IPMI: hopefully a climax
Lessons
Future Work

Analysis of other vendors’ implementations
Dell, HP, Lenovo, Oracle, etc.

Firmware update exploitation
Can an attacker inject a backdoor that persists?
Across BMC reboot? Across BMC flashes? Forever?

IPMI honeypot
Unclear whether attackers are exploiting these devices in the wild.
Some anecdotal evidence of their use as spambots.
Are they being used for other malicious purposes?
Conclusions

IPMI serves a vital role for system management. Carries elevated risks, potential for powerful attacks. At least some vendors are getting it badly wrong. Farmer is correct: IPMI is a serious concern. Our work: A call to arms.
IPMI: Understanding Your Server’s Remote Backdoor

Anthony J. Bonkoski
abonkosk@umich.edu

SUMIT 2013
## Zmap Scan Details

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Identifying Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuperMicro</td>
<td>Subjects containing “<a href="mailto:linda.wu@supermicro.com">linda.wu@supermicro.com</a>” or “<a href="mailto:doris@aten.com.tw">doris@aten.com.tw</a>”</td>
</tr>
<tr>
<td>Dell</td>
<td>Subject containing iDRAC</td>
</tr>
<tr>
<td>HP</td>
<td>Subjects containing “CN=ILO” and issuers containing “iLO3 Default Issuer” or “Hewlett Packard”</td>
</tr>
</tbody>
</table>

*Landing pages spot-checked for false positives*