Illuminating the Security Issues Surrounding Lights-Out Server Management

Anthony J. Bonkoski
University of Michigan
abonkosk@umich.edu

Russ Bielawski
University of Michigan
jbielaws@umich.edu

J. Alex Halderman
University of Michigan
jhalderm@umich.edu

Abstract

Out-of-band, lights-out management has become a standard feature on many servers, but while this technology can be a boon for system administrators, it also presents a new and interesting vector for attack. This paper examines the security implications of the Intelligent Platform Management Interface (IPMI), which is implemented on server motherboards using an embedded Baseboard Management Controller (BMC). We consider the threats posed by an incorrectly implemented IPMI and present evidence that IPMI vulnerabilities may be widespread. We analyze a major OEM's IPMI implementation and discover that it is riddled with textbook vulnerabilities, some of which would allow a remote attacker to gain root access to the BMC and potentially take control of the host system. Using data from Internet-wide scans, we find that there are at least 100,000 IPMI-enabled servers (across three large vendors) running on publicly accessible IP addresses, contrary to recommended best practice. Finally, we suggest defensive strategies for servers currently deployed and propose avenues for future work.

1 Introduction and Roadmap

The Intelligent Platform Management Interface (IPMI) is a standard for out-of-band system management that allows operators to remotely administer machines at a layer below the host system's CPU and software [15]. Modern IPMI implementations let administrators remotely monitor the health of the hardware, control the system's power state, attach virtual boot media, and redirect the keyboard, video, and mouse. All this functionality can be exercised remotely over an IP network, typically with either a command line interface or a web-based front end [12, 21, 24]. Major server OEMs each have a special name for their IPMI implementation, such as HP's iLO, Dell's iDRAC, Oracle's iLOM, and Lenovo's IMM.

The core of an IPMI implementation is the Baseboard Management Controller (BMC), an embedded microcontroller that is integrated into the system's motherboard or installed via a daughter card. The BMC has its own flash storage and runs its own operating system, separate from the host's. It typically has access to the PCI bus, to the on-board NIC via a "side-band" interface, and to a collection of sensors and I/O ports [24]. Consistent with its purpose, the BMC has almost total control of the server.

IPMI can be a convenient administrative tool, but, under the control of attackers, it can also serve as a powerful backdoor. Attackers who take control of the BMC can use it to attack the host system and network in a variety of ways. For example, they could install BMC-resident spyware to capture administrative passwords when the operator remotely accesses the host. They could use the remote physical console to boot the host into a recovery mode and gain root access, or they could use boot media redirection to run a separate OS and obtain raw access to the disks. Malware residing on the BMC could be extremely difficult to detect, since it sits at an even lower architectural layer than a BIOS- [9] or VM-based rootkit [17, 22], and it would survive reinstallation of the host OS or even complete replacement of the host's storage devices. We survey risks from compromised IPMI devices in Section 3.

Given these risks, one might assume that IPMI developers exercise rigorous security precautions to protect the BMC from remote compromise. To test this, we analyzed an IPMI implementation shipped by one large server manufacturer, Supermicro, which is based on firmware initially developed by ATEN Technologies. We find that the firmware contains numerous textbook security flaws, including exploitable privilege escalation, shell injection, and buffer overflow vulnerabilities. We demonstrate a proof-of-concept exploit against one of these problems—a buffer overflow in the web interface's login page—and show that it can be used to remotely obtain a root shell on the BMC. We describe our analysis in detail in Section 4.

Since BMC compromise is so dangerous, it has become a recommended practice not to connect IPMI devices to

public networks. Instead, security best practice calls for maintaining a physically isolated management network or at least a separate management VLAN [11, 25]. Unfortunately, we find that many server operators do not follow these recommendations. In Section 5, we use data from Internet-wide surveys to reveal public IP addresses of over 100,000 IPMI devices, including more than 40,000 systems that our results suggest are remotely exploitable.

In Section 6, we attempt to draw lessons from IPMI security failures and suggest mitigations for developers and users. The vulnerabilities we find, along with others previously found by Farmer [7] and Moore [19], suggest that some IPMI manufacturers are systematically failing to properly secure these devices and do not fully appreciate the security implications of out-of-band management.

These problems are compounded because although many IPMI implementations are based on GNU/Linux, they are ultimately closed systems that present a minimal interface compared to a general purpose OS. As a result, system administrators have little ability to inspect or control the internal operations. This prevents server operators from applying existing defensive security tools (e.g. Tripwire [16]) and complicates independent security analysis. Even if problems are found and firmware updates released, these updates typically need to be applied manually. Many server operators may not realize how important it is to keep their IPMI firmware up to date and their management interfaces off of the public Internet.

2 Related Work

Until recently, little work has focused on IPMI security issues, and research in this area remains in its infancy. Dan Farmer was one of the first security researchers to develop a deep interest in IPMI risks and published a discussion of the technology and the potential threats it presents in early 2013 [8]. Farmer also recently discovered an undocumented debugging feature in Dell's iDrac express 6 firmware that would let any user gain access to an SSH root shell [7]. Our work builds on Farmer's and exposes other serious problems with IPMI. We also demonstrate an even more dangerous attack against another vendor's implementation.

Recently, HD Moore disclosed multiple vulnerabilities in two of the most popular uPnP libraries [18]. Among the affected devices was the Supermicro IPMI implementation we analyze in this study. Moore et al. developed a Metasploit module that exploits one of the vulnerabilities and targets ATEN-based Supermicro IPMI systems [19]. Moore reported finding approximately 35,000 such systems publicly accessible and exploitable.

In general, malware targeting non-PC devices is a growing trend. One example is malware infecting home routers, which are architecturally reminiscent of the IPMI implementations we studied. Both kinds of devices are low-power embedded systems that frequently run Linux, often expose web-based management interfaces on public IP addresses, and can be leveraged to attack traditional PCs connected to them. ISE recently released a detailed report on security issues with common home and office routers [14]. They suggest that almost every device they examined had some critical security flaw, and they claim 38 independent router-related CVEs. We conjecture that architecturally similar IPMI devices may suffer from similar flaws due to poor engineering and lack of security testing, which poses a significant threat since IPMI-equipped servers are more likely to be high value targets than home networks.

Work by Novak et al. discussed the security issues raised by remote management software, focusing on a commercial product called Absolute Manage [20]. Like the IPMI device we studied, that tool turned out to be riddled with blatant vulnerabilities, and, since it was designed to perform powerful management functions, an attacker could gain full administrator privileges through its exploitation. Novak et al.'s conclusion is similar to ours: remote management tools are a particularly risky class of systems and must be designed and implemented with careful attention to security.

BMC malware can be compared to other malware that resides at low levels of the computing stack. Malicious BIOS firmware received substantial attention in the early 2000s [4, 9]. If an attacker can compromise a system's BIOS, he can insert a backdoor that persists for the life of the machine and is difficult to detect or remove. IPMI malware carries similar threats and is likely easier to develop, since many BMCs run a standard operating system. BMC malware would also likely be easier to install remotely, due to IPMI's substantial network-facing attack surface.

3 IPMI Security Risks

IPMI's growing popularity and powerful capabilities make it a new and interesting attack vector. In this section, we discuss features of typical IPMI implementations that lead to heightened security risks, and we consider what an attacker might to after compromising a vulnerable BMC.

3.1 Attack Surface

IPMI devices tend to have a large attack surface. In the case of Supermicro's IPMI implementation, the BMC operates a web interface on TCP ports 80 and 443, a remote KVM console system on port 5900, a virtual boot media server on port 623, a system management architecture for server hardware (SMASH) command line processor over SSH on port 22, and an IPMI protocol interface on UDP port 623 [24].

To control access to these interfaces, IPMI deployment best practice calls for use of an isolated management network [25]. Whether this advice is emphasized in official documentation varies by manufacturer: HP explicitly recommends it [11], while Supermicro's user's guide provides detailed instructions on which firewall ports to open to allow remote connections [24]. Even if a management network is in use, an attacker might be able to connect to the BMC due to misconfiguration or breach of other network systems or by compromising the host system, so relying solely on a secondary network for security is insufficient.

Many manufacturers ship servers with IPMI enabled out of the box. In Supermicro's implementation, IPMI is turn on in the BIOS by default, and other default settings cause the BMC to obtain an IP address automatically via DHCP and to use either the dedicated management NIC *or* the primary onboard NIC in a failover configuration [24]. This greatly increases the risk that uses will inadvertently leave the BMC exposed on a public IP address. We note that IPMI implementations typically provide remote management capabilities even when the host system is shut down, so if IPMI is enabled the only way to eliminate the risk of attack is to unplug the system from the network or power supply.

3.2 Authentication Risks

All IPMI devices support basic authentication via usernames and passwords [15]. Several manufacturers ship devices with default administrative credentials, which the system administrator may neglect to change. Dell's iDRAC IPMI has used default credentials listed directly in its user manual (`root/calvin`) [5], as does the Supermicro system we analyzed (`ADMIN/ADMIN`) [24]. The Supermicro system also provides an undocumented Anonymous user account that is enabled by default and configured as an administrator [2].

Even if the user sets a strong password, it may be exposed through insecure storage on the IPMI device. The IPMI specification lists requirements for storing passwords in plaintext [15], and we confirmed that the Supermicro device we tested stores all passwords as plaintext in a single file (`PSBlock`) within the system's nonvolatile storage area. Administrators who operate a large number of servers may reuse the same passwords across multiple systems. Thus, compromising a single IPMI device might give an attacker access to many machines.

3.3 Attack Scenarios

Attackers could subvert the BMC by guessing default passwords, exploiting vulnerabilities, or flashing malicious firmware. These lead to a number of dangerous attack scenarios.

Subverting the host system An attacker with access to the IPMI device can take advantage of its remote management facilities to attack the host system or other machines on the management network. Typical IPMI implementations provide a remote virtual console, redirecting the keyboard, video, mouse, and serial port over the network. Another common feature is virtual USB disk media, which can be used to infiltrate or exfiltrate files or to provide new boot media. The combination of these capabilities and remote power cycling would allow an attacker to seize control of most common server configurations. For instance, they could restart the system and boot from a virtual live CD, then directly copy or modify data on the host's storage devices.

BMC spyware If the attacker can install malware on the BMC, it would have a powerful vantage point for spying on the system and its administrator. BMC spyware could eavesdrop on remote management sessions, sniffing passwords for the host machine and other network systems accessed from it. It could also potentially eavesdrop on the physical server console via IPMI's remote KVM functionality.

Persistent BMC rootkits As the BMC operates independently from the host's operating system and CPU, it provides an ideal hiding place for a stealthy, highly persistent rootkit. A BMC rootkit could provide the attacker with backdoor access that is hidden from IPMI access logs and insusceptible to password changes. A BMC rootkit would survive reinstallation of the host's OS, or even complete replacement of the host's storage devices. Such rootkits could even be designed to survive BMC firmware updates by dynamically patching the new firmware.

Attacking the BMC from the host system An attacker who compromises the host system could use it to attempt to compromise the BMC. With the Supermicro device we tested, software running on the host can reflash the BMC's firmware via a KCS (keyboard-controller style) interface, without any authentication or code signing [24]. Due to the closed nature of IPMI implementations, once attackers gain control of the BMC, it may be extremely difficult to detect their presence or remove them from the system.

IPMI botnets If widely used IPMI devices can be compromised remotely, they can be leveraged to create large networks of bots. This is an attractive attack, because although the BMC has limited processing power, most servers have substantially more network bandwidth than typical home PCs. Furthermore, the system operator is unable to run normal malware detection and removal tools within the BMC, so IPMI bots may have longer lives that their desktop equivalents. There have already been anecdotal reports about IPMI devices being used for this purpose in the wild [3].

4 Analysis and Attacks

To explore the potential for BMC compromise, we analyzed an IPMI implementation shipped by one large server manufacturer, Supermicro. This process involved examining firmware binaries obtained from the company's website and performing exploratory probing using a server we purchased. We ultimately discovered a range of vulnerabilities, and we developed two proof-of-concept exploits to demonstrate some of the most critical problems.

The server we experimented with is a Supermicro SYS-5017C-LF 1U rackmount system with a Super X9SCL-F motherboard. The server's BMC firmware was created by ATEN Technology Inc., which also supplies IPMI systems to other system vendors, and apparently customized by Supermicro. The firmware runs on a Nuvoton WPCM450 BMC integrated into the motherboard. Our server shipped with firmware version 1.86 for the X9 motherboard line, which is the most recent revision and dates to November 2012. Internally, the BMC uses an ARM926EJ-S CPU and runs Linux 2.6.17.

This IPMI device provides both a web-based front end and an SSH interface with a SMASH command line processor. We focused our investigation on the web interface.

Our analysis began with examination of the firmware image file. Using `binwalk` [10], we found that it contains two CramFS filesystem partitions and a compressed Linux kernel. The first partition contains the root-level mount point and the second contains the web resources, including HTML, JavaScript, and CGI (Common Gateway Interface) programs written in C. At boot, the BMC's kernel mounts these partitions in read-only mode. It also mounts a memory-backed `/tmp` partition and a 1.3 MB flash-backed `/nv` partition, which is mounted in read-write mode and used to store configuration and log files.

We proceeded to investigate the security of the JavaScript and CGI programs through a combination of code inspection, disassembly, and experimentation. This led us to uncover a series of vulnerabilities.

4.1 Insecure Input Validation

An insecure design pattern that runs throughout the ATEN-based web interface is that it appears to only perform input validation in client-side JavaScript and HTML, without any corresponding validation in the server-side CGI programs. This is dangerous, of course, because the attacker can modify or bypass the client-side checks to send arbitrary data to the server.

In every instance we examined, input size checking occurs entirely on the client. For example, on the login page, the only input size validation on the username and password fields is the text field limit set in the HTML. The server-side `login.cgi` program that receives this input does not perform size checks before performing `strcpy`, resulting in a buffer overflow. We further explore the implications of this vulnerability in Section 4.3.

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

Figure 1: **Bad Privilege Checking** — The Supermicro IPMI web interface checks user privileges with client-side JavaScript, without corresponding server-side checks.

Similarly, input sanitization appears to happen either in client-side code or not at all. This is especially problematic because several web page text fields present front ends to shell commands. In these instances, the server concatenates the input text with other parts of a Linux shell command and executes them using the libc `system` function. In many cases, no checks are performed in either JavaScript or the back-end code, even for easily validated formats such as IP addresses. This flaw leads to multiple shell injection vulnerabilities, one of which we exploit in Section 4.2.

The insecure client-side validation pattern applies not only to input sanitization but also to user privilege checks. The web interface manages user permissions on the client side by initiating an AJAX request to request the current user's permissions from the server and then calling a context-specific JavaScript function called `PrivilegeCallback` that is provided by the current page. One implementation of this function is shown in Figure 1. This appears to be the full extent of the device's privilege validation; the server does not further verify the user's permissions when handling a request. This allows any IPMI user to escalate permissions to gain full administrator access.

Figure 2: **Shell Injection Vulnerability** — The web interface fails to sanitize inputs that are directly used in shell commands. Here, the code in backticks gets executed.

4.2 Shell Injection Vulnerabilities

The lack of input sanitization leads to the potential for shell-injection vulnerabilities in several functions. Out of 67 CGI programs, we found 15 that call the `system` function. While we did not check whether all 15 are vulnerable to shell injection, this provides an upper bound. We did confirm that the CGI program responsible for updating the date and time (`config_date_time.cgi`) has a vulnerability in the IP address field used for NTP time updates. An example of a shell injection that executes the command "`sleep 60`" is shown in Figure 2.

The firmware ships with several commands that can be used to construct useful shell-injection payloads. HD Moore used the included `openssl` utility to implement a connect-back shell as part of his uPnP exploit [19]. Our approach used `wget` to retrieve code from another server and execute it. We piped the output to the system log file, which we could then retrieve with `system_log.cgi`. To ease the command injection, we also wrote a psuedo-terminal in Python to abstract away these HTTP requests. This approach gave us an indirect root shell on the BMC, allowing us to explore the system's operation from the inside.

This shell-injection exploit was useful for our analysis because it gave us a beachhead through which to explore the running server from the inside. The attack requires an IPMI user account, but this is still extremely dangerous if, for example, the user has not changed the default login credentials. Even without an account, an attacker can still gain root access to the BMC by exploiting another vulnerability, which we discuss below.

4.3 Buffer Overflow Vulnerabilities

There are numerous buffer-overflow vulnerabilities in the web interface's CGI programs due to lack of input validation and bounds checking. One such example is `login.cgi`, which uses the unsafe `strcpy` function to manipulate user-controlled inputs. Figure 3 presents a partial decompilation of the vulnerable code. The `cgiGetVariable` function returns a pointer to a buffer containing the requested CGI variable. As the listing shows, this string, which is of unconstrained size, gets copied into a fixed-size buffer without any length checking, so long user inputs will overwrite the contents of the stack. Thus, Supermicro's ATEN-based IPMI web

```
int main(void)
{
    char name[128], pwd[24];
    char *temp;

    // ... initialize ...

    temp = cgiGetVariable("name");
    strcpy(name, temp);
    temp = cgiGetVariable("pwd");
    strcpy(pwd, temp);

    // ... validate user ...
}
```

Figure 3: **Exploitable Buffer Overflows in login.cgi** — The Supermicro IPMI web interface uses this server-side code to handle HTTP POST requests from its login page.

interface has buffer overflows in its login page's username and password fields.

Many of the other CGI programs appear to have the same kind of vulnerability. More generally, the pattern in Figure 3 is an idiom the developers seem to have used everywhere they handle POST requests: they use a call to `cgiGetVariable` and then `strcpy` to a fixed-size stack-based buffer. All these instances are potentially vulnerable to buffer-overflow attacks.

There remains the question of how exploitable these vulnerabilities are. In particular, does the BMC employ modern buffer-overflow defenses, such as DEP, ASLR, and stack canaries? Using a combination of shell-injection and disassembly, we determined that neither DEP nor stack canaries are in use. There is a limited ASLR implementation, but it only randomizes the location of the stack and heap; all libraries are deterministically mapped. We verified this configuration for DEP and ASLR by examining the memory maps of processes. The stack always had `rwxp` permissions on its pages, and shared libraries were always mapped to the same locations across various executions while the stack base address varied.

4.4 Buffer Overflow Exploit

We created a proof-of-concept exploit for the vulnerable `login.cgi`. To ease development, we leveraged the root shell access gained via the shell-injection attack described above to install a modified `ssh-daemon` that forks a normal shell instead of the limited SMASH command interpreter. We enabled core dumps and installed a cross-compiled `gdb` to analyze them. We also temporarily disabled ASLR until our basic exploit was working.

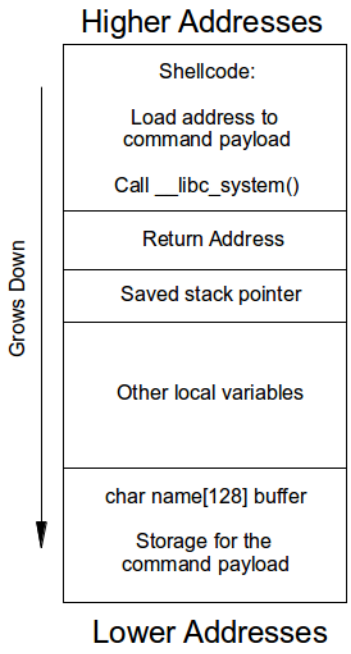

Figure 4: **Exploit Memory Layout** — Our proof-of-concept exploit overflows the buffer `name[128]` onto the stack. We reuse the allocated part of `name` to store a shell command, which we execute with libc's `system()`.

Since the system does not use DEP, we chose to implement a traditional stack-executed attack. That is, we placed specially chosen ARM instructions on the stack via the overflow and set the return address to jump to the stack [1]. Our goal was to execute a shell command via `system`; however, there were few bytes left for the command payload after the shellcode, so we placed the command in the allocated part of the `name` buffer. To prevent the program from crashing before the function returns, we have to ensure that one local variable (a pointer to a structure in a shared library) remains intact, but this value appears to be constant in practice. Figure 4 depicts our exploit's memory layout.

As a simple example payload, we decided to download and launch a modified `ssh-daemon` that forks a root shell when the *incorrect* password is entered. This modification required changing only two instructions in the system's original SSH daemon.

Lastly, we had to overcome the randomized stack. This proved simpler than one might suspect. ASLR is relatively weak with a 32-bit address space [23]. The lower 12 bits are fixed due to 4 KB page alignment, and several high-order bits are constrained by other reserved memory regions (such as text, heap, and libraries). On the IPMI device, we found that only 12 bits (12–23) were being randomized, yielding a mere 4096 possibilities.

This search space is fairly easy to brute-force. However, since the device uses a low-powered embedded processor, it has difficulty handling continuous web requests. We found that sending requests at intervals of around 200 ms was tolerated by the system. At this rate, our exploit succeeds within about 7 minutes on average.

It may be possible to develop an exploit that succeeds in a single request by using a return-to-libc attack. However, this approach is complicated because the stack pointer is mangled during the overflow, and ARM calling conventions pass parameters in registers.

Even without further optimization, our exploit is extremely dangerous because it can easily be parallelized to attack many servers at once. We conservatively estimate that it would take less than an hour to launch successful parallel attacks against all of the 40,000 ATEN-based Supermicro IPMI devices that we observed listening on public IP addresses (see Section 5).

4.5 Vulnerable Models

To understand how widespread vulnerabilities like these are across the Supermicro server product line, we downloaded the current set of IPMI firmware images available on the company's support site[1] as of May 23, 2013. Out of 64 distinct firmware images, 30 appeared to use ATEN-based software very similar to the implementation we tested. We disassembled the `login.cgi` program from each of these images, and all of them appear to contain similar buffer overflow vulnerabilities. These vulnerable firmware images apply to 135 Supermicro product models. The problems may also affect IPMI devices from other manufacturers that are based on similar ATEN firmware.

5 Network Measurements

Given the security risks of IPMI devices, best practice dictates that they should not be accessible from the Internet. One possible explanation for the widespread vulnerabilities in the ATEN-based Supermicro implementation is that the programmers assumed uses would follow this advice and not connect the devices to public networks. Is this a safe assumption?

In order to estimate the number of publicly accessible IPMI devices, we used data from an Internet-wide network survey conducted in May 2013 using the ZMap network scanner [6]. This dataset includes the X.509 certificates presented by all HTTPS servers in the public IPv4 address space listening on TCP port 443. We searched the data for certificates that had the identifying characteristics of default HTTPS certificates used by the web interfaces for Supermicro, Dell, and HP IPMI implementations.

[1] http://supermicro.com/support/bios/firmware0.aspx

For Supermicro devices, we looked for certificates with subjects containing "linda.wu@supermicro.com" or "doris@aten.com.tw", which appeared in certificates from different versions of the firmware. For Dell devices, we looked for the string "iDRAC" in the certificate subject. For HP devices, we looked for subjects containing "CN=ILO" and issuers containing "iLO3 Default Issuer" or "Hewlett Packard". We spot-checked the landing pages these servers displayed to guard against false positives. Here are the device counts we found:

Platform	Devices on Public IPs
Supermicro IPMI	41,545
Dell iDARC	40,413
HP iLO	23,376
Total	105,334

These data show that at least tens of thousands of servers with IPMI are immediately at risk, and they may be only the tip of the iceberg. Other versions of the same firmware may have different certificate formats not included in these totals, and security-aware server operators may have generated non-default HTTPS certificates that would not match our search patterns. Our figures here are a lower bound on the number of IPMI-enabled devices exposed on public IP addresses today.

6 Defenses and Lessons

The problems we uncovered carry lessons for server operators, system manufacturers, and IPMI developers.

For server operators, the most practical immediate defenses are to keep IPMI firmware up to date, change default passwords, and *never* configure IPMI devices on public IP addresses. These devices should be isolated either on a physically separate management network or on a management VLAN [25]. Operators who do not need IPMI should disable it entirely if possible. Although these steps should already be considered security best practices, the large number of IPMI devices currently listening on public IPs suggests that many server operators are either unaware that their devices are publicly reachable or oblivious to the risks.

For IPMI developers and server OEMs, our findings should be a wakeup call. Given the power that IPMI provides, the blatant textbook vulnerabilities we found in a widely used implementation suggest either incompetence or indifference towards customers' security. While some OEMs recommend helpful precautions such as dedicated management networks [13], this should not be an excuse to shift blame to users who fail to heed this advice and suffer damage because of vulnerabilities in IPMI firmware. We believe that properly securing IPMI will require OEMs to take a defense-in-depth approach that combines hardening the implementations with encouraging users to properly isolate the devices.

Securing IPMI will require security expertise on the part of developers and careful scrutiny during system design, engineering, and testing. A starting point would be to adopt standard defense mechanisms, such as password salting and hashing, automatic firmware updates protected by digital signatures, and the use of DEP, ASLR, and stack canaries. Implementations should also be examined by qualified penetration testers. Even then, with such a large attack surface, vulnerabilities are bound to slip through, but they will likely be more difficult to find and exploit.

There are a variety of special-purpose security features that IPMI implementers should consider adding in future firmware. The most basic is to ensure that IPMI is disabled until explicitly turned on by the user. Another proactive security mechanism would be to have the BMC periodically check that it has not been accidentally attached to a public network, perhaps by requesting that a server operated by the vendor attempt to connect to it. If the connection is successful, the BMC could temporarily disable itself and alert the operator.

The problems we found may represent a kind of "impedance mismatch" between the server community and the low-power embedded systems community. Server owners are used to dealing with publicly accessible machines and have come to expect their systems to be designed for the rigors of the Internet, while embedded designers have long enjoyed the luxury of narrow use-cases and isolated systems. These vantage points must find synchrony. If these management systems are to be connected—even indirectly—to public networks, IPMI devices must be engineered with the same security scrutiny as traditional server systems.

7 Future Work

Research into the security of fielded IPMI devices is still at an early stage, and there are a number of promising avenues for future work.

Analysis of other implementations Since our study focused on one IPMI implementation from a single vendor, we can only draw limited broader conclusions from the vulnerabilities we found. They highlight potential risks, but they do not prove that poor security engineering is widespread in this class of devices. Further study is needed to analyze IPMI products from other major vendors, such as Dell, HP, Lenovo, and Oracle. Problems that occur across many implementations might suggest broader lessons or point to underlying root causes, and would help establish the true scope of IPMI threats.

Firmware update exploitation The attacks we investigated work by exploiting vulnerabilities in the BMC's web interface, but firmware updates offer a separate and interesting attack vector to explore. The BMC needs to provide a secure and reliable update mechanism for itself, and the implementations we surveyed all support updates both via the web front-end and using an out-of-band process from the host machine's OS. Only some vendors provide signatures for firmware updates [11, 5], and these are not always automatically verified. We encourage vendors to develop a more robust update process that ensures that firmware updates nominally intended to prevent against possible compromises do not become a point of weakness themselves.

IPMI honeypots It is unknown whether IPMI vulnerabilities like the ones we uncovered are being intentionally exploited in the wild. While there is anecdotal evidence that some BMCs have been turned into spambots [3], it is unclear whether attackers are specifically targeting BMCs or whether their simplistic vulnerabilities have allowed automated attack systems to compromise what would otherwise appear to be underpowered machines. We would like to address this question in future research by establishing IPMI honeypots that are instrumented to record evidence of attempted and successful attacks being launched against these devices.

8 Conclusion

Out-of-band management is a technology of great value to the IT community, but its benefits are accompanied by significant security risks. IPMI's remote administration features can be powerful tools in the hands of an attacker, and implementations tend to have large remotely accessible attack surfaces. Since BMCs operate independently of the host system and CPU, cleverly written malware running there could potentially reside undetected indefinitely. Unfortunately, due to the closed nature of BMC firmware, server operators have few avenues to defend themselves without vendor assistance.

To shed light on these risks, we analyzed the security of one IPMI implementation, the ATEN-based Supermicro BMC. We uncovered a wide range of vulnerabilities and demonstrated two working attacks that allowed us to gain root shell access. These problems pose an immediate threat to many systems in the field; we found over 40,000 devices similar to the one we analyzed visible on public IP addresses. We have disclosed these vulnerabilities to ATEN and Supermicro, and we hope they will provide firmware updates to fix the immediate problems. In the meantime, we urge all IPMI users to ensure that their management interfaces are not accessible from the Internet.

In the long run, securing remote management systems calls for a defense-in-depth approach. Vendors need to apply careful security engineering practices, minimize attack surfaces, and help users ensure that their systems are appropriately locked down and isolated from public networks. Unfortunately, our findings suggest that many users and at least some IPMI vendors are unaware of the security risks that out-of-band management entails. We hope research like this that exposes vulnerabilities in real implementations will lead to greater awareness and understanding of those risks and coordinated efforts to reduce them.

Acknowledgments

We thank Zakir Durumeric for providing Internet-wide scan data. We also thank Eric Wustrow and Pat Pannuto for their feedback and assistance. We are grateful to HD Moore and the anonymous reviewers for their insightful suggestions and comments. This work was funded in part by NSF grant CNS-1255153.

References

[1] Aleph One. Smashing the stack for fun and profit. *Phrack*, 7(49), August 1996.

[2] Floris Bos. Supermicro IPMI documentation omission: presence of second admin account. Full Disclosure mailing list, October 2011. http://seclists.org/fulldisclosure/2011/Oct/530.

[3] brc_csf. Supermicro IPMI security. Web Hosting Talk forum post, October 2010. http://www.webhostingtalk.com/showthread.php?t=992082.

[4] Michael Davis, Sean Bodmer, and Aaron LeMasters. *Hacking Exposed: Malware and Rootkits*. McGraw-Hill, 2009.

[5] Dell. *Integrated Dell Remote Access Controller 7 (iDRAC7) user's guide*, 1.30.30 edition, December 2012.

[6] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. ZMap: Fast Internet-wide scanning and its security applications. In *22nd USENIX Security Symposium*, August 2013.

[7] Dan Farmer. Dell backdoor, January 2013. http://fish2.com/ipmi/dell/secret.html.

[8] Dan Farmer. IPMI: Freight train to hell, January 2013. http://fish2.com/ipmi/itrain.html.

[9] John Heasman. Implementing and detecting an ACPI BIOS rootkit. Talk at Black Hat Europe, 2006. http://www.blackhat.com/presentations/bh-europe-06/bh-eu-06-Heasman.pdf.

[10] Craig Heffner. Binwalk: Firmware analysis tool. https://code.google.com/p/binwalk/.

[11] Hewlett-Packard. *HP Integrated Lights-Out security*, 7 edition, December 2010. http://bizsupport2.austin.hp.com/

bc/docs/support/SupportManual/c00212796/c00212796.pdf.

[12] Hewlett-Packard. *HP ProLiant Lights Out-100 User Guide*, March 2010. http://bizsupport1.austin.hp.com/bc/docs/support/SupportManual/c02063205/c02063205.pdf.

[13] Hewlett-Packard. *HP iLO 3 User Guide*, October 2012. http://bizsupport2.austin.hp.com/bc/docs/support/SupportManual/c02774507/c02774507.pdf.

[14] Independent Security Evaluators. Exploiting SOHO routers, April 2013. http://securityevaluators.com/content/case-studies/routers/soho_router_hacks.jsp.

[15] Intel, Hewlett-Packard, NEC, and Dell. *Intelligent Platform Management Interface Specification v2.0*, February 2004. http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/second-gen-interface-spec-v2-rev1-4.pdf.

[16] Gene H. Kim and Eugene H. Spafford. The design and implementation of tripwire: a file system integrity checker. In *2nd ACM Conference on Computer and Communications Security*, CCS '94, pages 18–29, 1994.

[17] Samuel T. King, Peter M. Chen, Yi-Min Wang, Chad Verbowski, Helen J. Wang, and Jacob R. Lorch. SubVirt: Implementing malware with virtual machines. In *27th IEEE Symposium on Security and Privacy*, SP '06, 2006.

[18] HD Moore. Security flaws in Universal Plug and Play: Unplug, don't play, January 2013. https://community.rapid7.com/docs/DOC-2150.

[19] HD Moore, Alex Eubanks, and Richard Harman. Metasploit module for uPnP attack on Supermicro IPMI devices, February 2013. https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/multi/upnp/libupnp_ssdp_overflow.rb.

[20] Jay Novak, Jonathan Stribley, Kenneth Meagher, and J. Alex Halderman. Absolute pwnage: Security risks of remote administration tools. In *15th International Financial Cryptography Conference (FC)*, February 2011.

[21] Weimin Pan and Haihong Zhuo. IPMI configuration on ninth-generation Dell PowerEdge servers. *Dell Power Solutions*, August 2006. http://www.dell.com/downloads/global/power/ps3q06-20050317-Zhuo.pdf.

[22] Joanna Rutkowska. Introducing Blue Pill. The Invisible Things Lab's blog, June 2006. http://theinvisiblethings.blogspot.com/2006/06/introducing-blue-pill.html.

[23] Hovav Shacham, Matthew Page, Ben Pfaff, Eu-Jin Goh, Nagendra Modadugu, and Dan Boneh. On the effectiveness of address-space randomization. In *11th ACM conference on Computer and Communications Security*, CCS '04, pages 298–307, 2004.

[24] Supermicro. *SMT IPMI User's Guide*, 2.1c edition, 2013. http://supermicro.com/manuals/other/SMT_IPMI_Manual.pdf.

[25] Johannes Ullrich. IPMI: Hacking servers that are turned "off". ISC Diary blog, June 2012. https://isc.sans.edu/diary/IPMI%3Aminimal+Hacking+servers+that+are+turned+%22off%22/13399.