dan farmer/zen@fish2.com
Version 1.0, April 15th, 2013

IPMI++ Security Best Practices

1. Introduction to IPMI++ Security

The **Intelligent Platform Management Interface** (aka IPMI) defines a de facto standard mechanism to manage servers using Out of Band (OOB) or Lights-Out. IPMI runs on the BMC, an embedded computer subsystem found on pretty much all server motherboards made in the last 10 or 15 years. Often found running Linux, the BMC's CPU, memory, storage, and network can run mostly independently from the rest of the server.

The BMC's OS runs network services (web, telnet, VNC, SMTP, etc.) to help manage, debug, monitor, reboot, and roll out servers, virtual systems, and supercomputers. Vendors frequently add features and rebrand OEM'd BMCs: Dell has iDRAC, Hewlett Packard iLO, IBM calls theirs IMM2, etc.; in an attempt to avoid confusion I'll simply dub them all IPMI++.

IPMI++ is popular because it helps raise efficiency and lower costs associated with availability, personnel, scaling, power, cooling, and more. It also has a lot of options on how it's operated, setup, and configured, and just like other complex systems some of the paths you can take are less desirable than others, at least security-wise.

Security rarely has black-and-white delineations between what is right and wrong – the context, which includes the threats and the risks of your environment are what really matter, and ultimately you have to make the decision as to which path to take in order to defend your assets.

So in the absence of any context, I'll say what I *personally* consider to be ill advised; if at all possible or necessary I'll try to explain as well. I wrote an overall assessment on IPMI security that might prove useful for further context. Most of the more serious security issues covered in that paper can't be remedied by simple configuration settings, but it doesn't mean these are meaningless.

I'll place quotes in italics; when not explicitly cited you may assume the source it will be from the IPMI 2.0 specification, which may be found:

http://www.intel.com/design/servers/ipmi/spec.htm

Occasionally I'll highlight a few important bits in red.

It should be noted that this is a security document, not an operations or cost analysis one. It's possible that some suggestions here would be expensive to implement and manage; it's up to each organization to decide what the threat is if the advice isn't followed and if you're willing to absorb or manage the risk.

I'll start going over some assumptions, move to network and architectural issues, and then finally cover IPMI and BMC security. If possible I'll note the FreeIPMI configuration value that is returned by the named configuration tool (e.g. *bmc-config*, *pef-config*, etc.) and what the value should be. These tools allow you to manage the configuration values that govern at least some of IPMI security and operations of a BMC.

2. Some Lemmata

There shouldn't be much substantive disagreement about these, but nevertheless they seem to be a source of some incorrect security assertions and postures.

Administrative rights on a server (e.g. root, administrator, whatever appropriate for the OS) give the power to:

1) Manage IPMI accounts[1], including adding, removing, modifying any aspect of IPMI accounts on the local server.
2) Manage (start, stop, configure, etc.) all aspects of applications and services that can be run on the BMC. A few of these are mentioned in the IPMI specification, but most are vendor add-ons (SSH, the web UI, virtual media, etc.)
3) The Ethernet interface and the IP address that IPMI listens to may be changed via the web or command line tools. This includes forcing it to share the same physical Ethernet jack(s) as the server.

If you believe these, then any server that is compromised puts its BMC in jeopardy of compromise as well.

Turn on a disabled IPMI subsystem if IPMI is not currently enabled[2].XXXX

3. Basic Operations, Security & Architecture

There's not a lot you can do to secure the actual BMC, so you want to work around its limitations with strong network architecture and monitoring.

If you use a web interface to interact with the BMC/IPMI always use the SSL interface (e.g. *https* or port 443.) Be aware – if anyone can get on your management network they will probably be able to grab your passwords even though you use SSL, due to ARP spoofing and man-in-the-middle attacks[3]. Pure IPMI 2.0 commands using a good cipher are reasonably safe from network sniffing, assuming you have a strong password (see the next section for more on this.)

1. **Severely restrict any network access to any BMC** as well as the BMC's capability for outbound communications; this has to be done at the network layer (e.g. routers, switches, network devices, etc.), since the BMC has no network defenses or firewall capabilities.

   To be clear: **never let any network traffic from outside world** (e.g. those not on the management network zone) **touch or even breathe on any scrap of your BMC's active IPMI network interface** – no Serial over LAN, web interface, the IPMI protocol (UDP 623), no nothing.

   Finally, BMCs have small but mighty processors, but they will go down if they're subject to a DDOS/DOS attack. Keep them away from the enemy!

2. Restrict and alarm outbound network traffic and access for the BMCs – unless you work for Google, your BMCs should have no reason to talk to google.com et al. If a BMC is compromised it will probably want to talk to the outside world, this should be an easy thing to catch.

---

[1] There is almost always an anonymous login (see section 6.9.1 of the IPMI 2.0 specification) with User ID 1 that cannot be removed.

[2] xxxx

[3] Moxie Marlinspike has a nice write up and video on this topic, along with his SSLstrip tool.

3. If possible keep all IPMI network interfaces on their own segregated network. The IPMI managed network segregation should optimally be physical, but even having the interfaces on a separate VLAN is better than nothing (unless, of course, the added complexity of managing the separation causes more problems than it tries to solve.)
4. If the BMC can't have or doesn't support a dedicated Ethernet connection, use VLANs (which aren't great for security, but it's at least a paper thin wall of added protection.)
5. Add a layer of by placing a very secure computer to serve as a bastion host between the management network and the unwashed masses of computers at large.
6. Two-factor authentication, with at least one of the two factors unique to the management network (e.g. single sign on from the corporate network should not grant access!), should be required to gain access to the network.
7. Work with the information security team to build a set of best practices and policies around BMC and IPMI security.
8. **Ensure that the BMC storage is wiped or reset, including passwords, when de-provisioning servers**. End-of-life or de-provisioning servers must be handled carefully, as IPMI passwords may be recovered from a BMC. I know of no vendor[4] who gives advice or standard on how to erase any stored passwords on a BMC, so consider shredding, melting, or otherwise nuking the BMC/motherboard. At the very least remove the accounts used, or set the passwords to something other than your own.
9. Develop processes and incident response capabilities for dealing with a compromised server that takes into account the complexities a BMC adds. Since you probably can't easily tell if the BMC has been compromised it'll be a guess as to whether or not the IPMI password was compromised.
10. No other computers should be using this managed network unless they are directly serving the managed network or BMCs running IPMI (for instance I've seen database, backup, and other critical servers sharing this network, since it has been assumed secure.)
11. Do not share network space with different IPMI management domains (for instance the web server IPMI group and the database IPMI group, which might be owned by two different sets of administrators and use different IPMI passwords.) Segregate management domains with firewalls or other network gear.
12. Monitor the traffic on the management network. There probably isn't a lot of activity going on there (with some notable exceptions) and anomalies might well be easier to spot than on the busy server network.
13. Treat the BMCs as real servers (they are!): ensure that they're monitored, scanned for vulnerabilities, have their logs go to logging servers, etc. Ensure that you have the capability to record or sniff management network traffic as well as run IDS or other security alarm systems.
14. You should **probably** keep up-to-date with the most recent firmware for your BMCs as you can, and even more so if the release notes mention that they fixed security issues. I only say probably because who knows what they do in those black boxes,

---

[4] I believe newer Dells has a way of blowing away all settings and passwords but have not confirmed how to do this.

and they may introduce new and worse problems. At the very least attempt to keep your firmware version numbers the same for as many servers as you can, to keep the vulnerabilities consistent (rather than inconsistent and unknown.)

4. IPMI Security & Configuration

Here I'll cover two basic types of security – configuration and process. I put them together because I think it's easier to understand.

There are so many different ways of doing these for various vendors I won't try enumerating all of them. While in theory quite a few could be checked with software checkers, there aren't a lot of choices out there (I've some small tools to detect and look for obvious problems at http://fish2.com/ipmi) at this time.

I do, however, go over a variety of FreeIPMI configuration variables and suggested values in the next section that should work for almost any version of IPMI.

Other than the very first item (#1) these aren't in order of importance or security.

1. **Never use Cipher zero (0)**. The importance of this cannot be overstated. Simply having cipher 0 enabled on a channel allows anyone to perform any IPMI action with no authentication, effectively *subverting IPMI security entirely*. **Disable it at all costs**.

   To add to this warning: f that didn't sound dire enough, many, if not most, vendors have this ENABLED BY DEFAULT. So really, turn it off.

2. If you can't disable cipher zero see if you can force the usage of version 1.5 of IPMI, which doesn't support ciphers.

3. Only use ciphers 3, 8 & 12. IPMI supports various algorithms for authentication, integrity, and confidentiality; why wouldn't you want all 3? The only ones that support this goal reasonably well are these ciphers. There's a nice table outlining this in the spec, section 22.15.2 - "Cipher Suite IDs" – that lists all the supported algorithms. If at all possible disable all the others on all channels.

4. The BMC should use its own dedicated Ethernet connection (e.g. don't share the server's physical connection!)

5. Use 16 or 20 character passwords[5] (or passphrases, if you ever want to remember them) if you're going to use shared and widely disseminated and passwords. BMC passwords can often be attacked by brute force (some do exponential back-off, some just keel over, but others allow fast checking.) There are many ways to get a password from a BMC; one of them is by cracking a hashed entry that's stored on the BMC. At least make it a bit challenging – even unsalted a 20-character pass phrase can be a bit challenging to crack.

6. Change your IPMI passwords on an ongoing basis. This is painful, I know, because of a paucity of cross-vendor or enterprise quality IPMI password managers. I would suggest at least once per year, or whenever someone who knows the passwords leaves the group or organization. It's difficult to audit such changes since IPMI has no support for tracking when a password has been changed, plus you can't easily try to crack the passwords. Use process and rigor to ensure this happens.

---

[5] If you're running an environment that still runs version 1.5 of IPMI you could reasonably use 16 character passwords.

7. Ensure the lifecycle of accounts and passwords on the BMC are managed appropriately. Monitor them to ensure that any changes are spotted; if new accounts appear, privileges change, etc. follow it up with an investigation. If you use automation on the BMCs and store the password(s) somewhere you can even test to see if they've been changed (for better or for good!) Come up with a plan to change passwords on an IPMI management group on an ongoing basis – it's not acceptable to leave them unchanged for long periods of time. If some systems fall through the cracks and aren't changed with the rest either manually change the password or keep a list of historical passwords and try those. IPMI deals with none of this, but it's pretty vital.
8. Don't allow accounts with a null username or password. Anonymous logins – part of the spec – should always be disabled.
    a. *6.9.1 'Anonymous Login' Convention*
    b. *The IPMI convention for enabling an 'anonymous' login is to configure the entry for User ID 1 with a null username (all zero's) and a null password (all zero's).*
9. Remove or disable other default vendor accounts, or at least give them strong passwords. I'd highly recommend disabling them, since it can be difficult testing the strength of an IPMI password.
10. Create and use your own accounts – don't use the default ones given by the vendor.
11. Use v2.0/RMCP+ RAKP Authentication if you're able to. The KG key should be set to something other than the default (all zeros.)

    This is the hardest recommendation to follow in this document and some of you out there may well laugh at the thought of even trying. RMCP+/RAKP authentication is pretty daunting at the best of times (and is hence rarely used), but it does offer one of the best defenses against IPMI's worst problem of widely shared and reusable passwords. RAKP authentication essentially provides a second password, and if used should be different for every BMC.
12. While it's possible to have different usernames, IDs, passwords, and more for each channel, that way leads to madness and confusion. Never do this unless you really know what you're doing or someone points a loaded gun to your head.
13. Note – it's acceptable (but perhaps not from an administrative or management standpoint) to have users with a special level on one channel and lower levels on others. Be aware that IPMI has no provisions or way of informing you of any changes made – the creation, deletion, or management of users and their permissions are not natively logged or available.
14. Don't allow the user to set the Session Inactivity Timeout to too long a value. A bit arbitrary, but 5 minutes seems more than enough (it defaults to 1 minute (according to "Table 6-7, Default Session Inactivity Timeout Intervals".)
15. Explicitly use IPMI version 2 (when possible) when using command line utilities such as *ipmitools*, *racadm*, etc. All the utilities I've seen have options (e.g. "*ipmitool -I lanplus …*") to do this. If you don't they generally use IPMI version 1.5, which doesn't use all that lovely encryption you've setup.
16. Don't allow or use OEM cryptography unless you're really, really sure it's better than what the specification already provides. Only trust published, peer-reviewed, and well-regarded cryptographic systems.

17. Don't use MD2 or RC4 for anything (they're usable in several places in the specification and vendors still support them.) Written in 1989, they've been both demonstrated to be relatively insecure. MD5 isn't great, but it's better than MD2.
18. Don't disable per-message and user level authentication.

    The spec says:

    a. *6.12.4 Per-Message and User Level Authentication Disables*
    b. *[…] In some cases however, the connection medium is considered to be trusted even though multiple user sessions are allowed. Once a session has been activated, the computational overhead of authenticating each packet may not be necessary.*

    Suck up the "overhead" and don't disable it.
19. Don't disable Link Authentication.

    One of those don't worry, be happy and be insecure things. My advice – don't be **too** happy.

    a. *6.12.5 Link Authentication*
    b. *Link Authentication is a global characteristic associated with the connection mode for the channel. Link Authentication is enabled/disabled via the serial/modem configuration parameters. When Link Authentication is enabled, it is necessary to identify one or more users that will serve as the source of the username (peer ID) and password information for the link. This is accomplished by setting an 'Enable User for Link Authentication' bit in the Set Channel Access command.*
    c. *For physically secure connections, these 'Link Authentication' protocols may be all that's considered needed to authenticate the user.*

    Don't listen to them. Who are you going to believe: me or your lying eyes?
20. Disable gratuitous ARP replies. An ARP is a packet defined in RFC 831 that permits computers to find a physical Ethernet (aka MAC) address and map it to an IP address. A gratuitous ARP reply is when a computer sends a broadcast network packet to update the mapping between an IP address and an Ethernet, or MAC, address. While gratuitous ARPS may be useful at times, BMCs shouldn't be sending traffic on the local LAN anyway, and it may be used to spoof addresses.
21. The BMC should use static IP #'s, not DHCP. You want to have control over where your BMC shows up: monitoring, security, maintenance, etc. all benefit.
22. Log any scrap of data that the BMC allows (not much, I know.)
23. Disable all services that aren't used (this can usually be done via the BMC's web interface, scripting interfaces, or the command line interface.) If I catch you using telnet I will beat you. Put the services or the BMC's configuration under proper change management and the appropriate processes for your organization. If there are both unencrypted and encrypted versions of a service disable the former and only allow the latter.

5. FreeIPMI configuration variables

FreeIPMI contains a widely used set of tools that among other things allows you to check and set a variety of IPMI settings on the BMC. These are the configuration values that I've been able to find that have some bearing on security.

Unfortunately the tools only run on Linux, but should work on remote systems no matter what operating system it runs, as it simply communicates via the IPMI network protocol.

Because version 1.5 of IPMI doesn't use encryption you could in theory, and arguably should, disable as many of the 1.5 features as possible, but that gets a bit tricky and I, for one, won't try that. If you can't disable cipher zero then using version 1.5 IPMI is actually more secure, since cipher zero was introduced in version 2.0.

Section (from *bmc-config* unless specified)	**Name**	**Value**
`Rmcpplus_Conf_Privilege`	`Maximum_Privilege_Cipher_Suite_Id_0`	The variable should not be present at all.
`Rmcpplus_Conf_Privilege`	`Maximum_Privilege_Cipher_Suite_Id_XX`	There are 16 cipher suites possible in the specification; bmc-config should not return any of them other than 3, 8 and 12.
`Lan_Conf_Security_Keys`	`K_G`	Non-zero value; This value alone doesn't mean that you're actually using RAKP authentication, but you can't use RAKP without a key, and an all-zero key is pretty worthless.
`Lan_Conf_Misc`	`Enable_Gratuitous_ARPs`	`No`
`Lan_Conf`	`IP_Address_Source`	`DHCP`
`PEF_Conf`	`Enable_PEF`	`Yes`
`PEF_Conf`	`Enable_PEF_Event_Messages`	`Yes`
`PEF_Conf`	`Enable_PEF_Alert_Startup_Delay`	`No`
`PEF_Conf`	`Enable_Alert_Action`	`Yes`
`PEF_Conf`	`Startup_Delay`	`0`
`PEF_Conf`	`Alert_Startup_Delay`	`0`
`Lan_Conf_User_Security`	`Bad_Password_Threshold`	Any value > 0
`Lan_Channel`	`Volatile_Enable_User_Level_Auth`	`Yes`

Lan_Channel	Volatile_Enable_User_Level_Auth	Yes
Lan_Channel	Volatile_Enable_Per_Message_Auth	Yes
Lan_Channel	Volatile_Enable_Pef_Alerting	Yes
Lan_Channel	Non_Volatile_Enable_User_Level_Auth	Yes
Lan_Channel	Non_Volatile_Enable_Per_Message_Auth	Yes
Lan_Channel	Non_Volatile_Enable_Pef_Alerting	Yes
Serial_Channel	Volatile_Enable_User_Level_Auth	Yes
Serial_Channel	Volatile_Enable_User_Level_Auth	Yes
Serial_Channel	Volatile_Enable_Per_Message_Auth	Yes
Serial_Channel	Volatile_Enable_Pef_Alerting	Yes
Serial_Channel	Non_Volatile_Enable_User_Level_Auth	Yes
Serial_Channel	Non_Volatile_Enable_Per_Message_Auth	Yes
Serial_Channel	Non_Volatile_Enable_Pef_Alerting	Yes
SOL_Conf	Force_SOL_Payload_Authentication	Yes
SOL_Conf	Force_SOL_Payload_Encryption	Yes
Community_String	Community_String	Don't use "public"!
Lan_Conf_Auth	Callback_Enable_Auth_Type_MD2	No
Lan_Conf_Auth	Callback_Enable_Auth_Type_OEM_Proprietary	No
Lan_Conf_Auth	User_Enable_Auth_Type_MD2	No
Lan_Conf_Auth	Admin_Enable_Auth_Type_MD2	No
Lan_Conf_Auth	OEM_Enable_Auth_Type_MD2	No
Lan_Conf_Auth	Operator_Enable_Auth_Type_MD2	No
Lan_Conf_Auth	Callback_Enable_Auth_Type_None	No
Lan_Conf_Auth	User_Enable_Auth_Type_None	No
Lan_Conf_Auth	Operator_Enable_Auth_Type_None	No
Lan_Conf_Auth	Admin_Enable_Auth_Type_None	No
Lan_Conf_Auth	OEM_Enable_Auth_Type_None	No
Lan_Conf_Auth	SOL_Payload_Access	Yes
Chassis_Boot_Flags	User_Password_Bypass	No
userXX (there are 16 different user sections, this is true for all.)	Username	If variable is defined, this should not be null
userXX	Serial_Enable_Link_Auth	Yes

6. Security issues with no fix or answer

Amazingly the specification provides a way of querying the BMC and finding out if there are any anonymous users – e.g. those with no username and password – are enabled. As it says:

> *6.9.2 Anonymous Login Status*
>
> *The Get Channel Authentication Capabilities command includes a 'Anonymous Login Status' field. This field indicates to a remote console application whether User ID 1 is presently configured with a null username and null password. In addition, a bit is provided that indicates whether there are also non-null usernames enabled for the channel, or whether User ID 1 holds a null username, but a non-null password.*
>
> *Together, these bits can be used to guide a remote application in presenting connection options to a user.*

Yes, I suppose an attacker needs a guide to remotely break in[6].

De-provisioning, as discussed in the operations section, is troublesome, since physical access to a server can allow an attacker to access your IPMI passwords. The vendors simply have to do something about this.

There are some things that are optional that vendors have hopefully implemented, such as Slot Stealing (6.12a) and others, but there's no way of knowing without some substantial legwork.

7. Meta note

Ensure that vendors stay away from reserved (as specified by the protocol) areas. There are various fields and bits specified in IPMI protocol that should not be mucked with – the specification specifically says that they should be 0 and not used. This is not only for future proofing (for instance IPMI 2.0 uses reserved bits in the 2.0 standard), but if they're being used something shenanigans might be taking place.

8. Thanks!

I'd like to extend big thanks to the folks on the IPMI tool and Free IPMI mailing lists for suggestions on how to improve this. Albert Chu, Andy Cress, and Jarrod B. Johnson (alphabetically) in particular were helpful. I should note that this was done with help from a contract in DARPA's fine Fast Track Security program, so thanks to them as well.

Of course all errors, omissions, or bad advice are mine alone.

9. Glossary

Some terms used and explanations.

BMC. An embedded computer on server motherboards, and was designed to facilitate out of band (OOB) operations and implement the Intelligent Platform Management Interface (IPMI), a standard created by Intel and a consortium of large server vendors.

Channels. A fair bit of IPMI security configuration deals with channels – conduits that allow communications over various types of media, virtual or real; in here I essentially

[6] I wrote a small Python program that sends a packet and interprets the response to help audit this.

treat them equivalently, and claim that they in general face similar security issues. There are those that claim for channels that are physically tied to a server – say, a serial console – that security isn't necessary (the IPMI specification discusses some of this); I disagree.

iDRAC/iLO/IMS/RSA/etc. Nearly all servers made today implement the IPMI specification. Server and firmware vendors add features and use a variety of different names; Dell calls theirs iDRAC, Hewlett Packard iLO, IBM IMM, etcetera, but in the end it's all IPMI under the hood.

IPMI (Intelligent Platform Management Interface.) A standard that defines a computer system interface used for out-of-band or lights-out server management. Defined by Intel and a host of other server vendors, nearly all servers manufactured in the world implement the standard to varying degrees of compliance. IPMI has been around since 1998 and is currently on version 2.0 of the specification.

IPMI versions. IPMI 1.5 was widely implemented and will still be found on various servers (1.0 is probably still running in some places, but I've never seen it); 2.0 was defined in 2004 and most servers now implement it, although they are also backwards compatible. 2.0's main security claim to fame was introducing cryptography to the protocols as well as the infamous protocol zero (0.)