System Management

– a security perspective

Leif Nixon

System updates

Should we ever update the system?

Some common update strategies:

1. If it works, don't touch it!
2. We pick and choose the most important updates.
3. We apply all updates.

System updates

Discovering the need to update

- Watch vendor announcement lists
- Watch general security lists
- Be part of a community that can afford to watch more specific resources

System updates

Triage – Do we care?

Useful scale:

1. Update later (scheduled or together with more important stuff)
2. Update as soon as we get an update in our preferred format
3. Take care of it *right now* in one way or another!

System updates

Triage – Do we care?

Caution: “Important” for the vendor may still mean “Critical” for you!

Most security advisories are not tuned for our environment, with many general shell users. A “local root exploit” is critical to us!

System updates

Updating the triage

Generally, things get more broken with time.

System updates

Updating the triage

Generally, things get more broken with time.

Sometimes: lucky breaks in the other direction;

“Ah, it turns out the bug is only triggered if you use feature X”

“We found this simple workaround”

System updates

Problems getting updates

- Waiting for repackagers
  Example: CentOS
- Waiting for vendor solutions
  Example: proprietary filesystem drivers
- Figuring out stuff built from source at site
  Example: a lot of big scientific packages

System updates

Solutions for getting updates

- *If it hurts, don't do it*
  Limit dependency on turn-key solutions, third-party dependencies as far as possible. Make vendors understand our pain. Again and again. Until they understand.
- *Prepare for building own packages*
  Example: For RHEL/CentOS set up mock environment, do test builds of likely/hard components (kernel, glibc, ...)

System updates

Solutions for getting updates

- Document source builds so they can be redone by others
  Automatic build scripts/systems may help...
  ...but not if too complicated!
- Make sure you have the credentials, licenses etc needed to download vendor updates when you need them.

Workarounds

Trust, but verify

Can we test that it works? Do we trust it?

Always remember that even if one *particular* exploit fails, you may still be vulnerable...

Workarounds

Systemtap to the rescue

Systemtap is often useful for working around kernel vulnerabilities.

Example: to block the CVE-2013-2094 vulnerability in the perfmonitor subsystem:

```
probe kernel.function("sys_perf_event_open")
  printf("sys_perf_event_open DENIED!\n");
  $attr_uptr = 0
}
```

How to deploy updates

Updating login nodes and system servers

- *Reboot not needed*

  Updates can often be applied right away. Make sure enough stuff is restarted! Do you know about `/usr/bin/needs-restarting`?

- *Reboot needed*

  Multiple login servers help. Multiple redundant system servers help too. Otherwise, make sure a quick reboot does not cause user problems (lost jobs, failed file operations, . . .)

How to deploy updates

Updating worker nodes

- *Reboot not needed*

  Updates may be applied right away, but...

  ...do you care about OS jitter? Can we do work in the job epilog?

  ...are you sure the update persists? (more on this later)

How to deploy updates

Updating worker nodes

- *Reboot needed*

  Automate rolling updates (rebooting as soon as current job is done)!

  Keep users out of nodes they do not run jobs on.

  Can we accept the risk that users can login to nodes that are not yet updated? If not, shut down user login access to nodes.

No workaround, no update?

Decide *beforehand* when it will be appropriate to shut down access. Get management acceptance for this.

No workaround, no update?

Decide *beforehand* when it will be appropriate to shut down access. Get management acceptance for this.

Levels:

1. No new logins accepted, but do not kick out those who are logged in.
2. No logins accepted, logged in users kicked out, jobs keep running.
3. Jobs killed, too.

No workaround, no update?

Keep users and management informed!

> *"We have temporarily blocked logins while we are investigating how to secure the system against a serious security vulnerability that was released today"*

easily becomes

> *"I dunno, I heard they shut down the system because it was hacked or something."*

Configuration management

A consistent system configuration over nodes and servers is important for security, but also for functionality and performance.

Configuration management

Node installation

Special consideration for compute nodes: When and how do we (re-)install them?

1. Manually installed node image, use that when booting
2. Scripted install (kickstart etc) for making node image, use that when booting
3. Scripted install when needed, reboot from disk otherwise
4. Scripted install on every boot

Configuration management

Node installation

Node reinstallation on every reboot:

Good: Nodes are always clean.

Bad: Nodes already zapped when you want to do forensics.

Configuration management

Deployment at scale

Letting thousands of servers wget/rsync from a few system servers might not work.

Solutions: Bittorrent? Multicast?

Configuration management

Keeping things consistent

How do we make sure we do not forget vital configuration?

1. Checklists
2. Scripts
3. Configuration management tools – Ansible, Cfengine, Chef, Puppet, Quattor, …

Configuration management

Keeping things consistent

How do we make sure we do not forget vital configuration?

1. Checklists
2. Scripts
3. Configuration management tools – Ansible, Cfengine, Chef, Puppet, Quattor, …

Warning: Abstracting too much may make system administration harder. Strive for the right balance!

Configuration management

Keeping things consistent

Then add version control on top of this. Now, you also know how it was four months ago!

Log your actions. Still useful, even with configuration management tools (but may partly be the commit log).

May be as simple as a date-ordered text file on the system server.

Configuration management

Keeping things consistent

Package your own tools, scripts etc.

Example: Instead of some slightly different scripts in /root/bin on four clusters, we might aim for versioned RPM package on internal repo server, with source in git.

Configuration management

Keeping things consistent

Package your own tools, scripts etc.

Example: Instead of some slightly different scripts in /root/bin on four clusters, we might aim for versioned RPM package on internal repo server, with source in git.

But again, don’t overdo this…

Configuration management

Node health checking

Running health checking scripts in prolog/epilog might not be all that security related, but it saves a lot of other trouble...

We can add test for “security problem X fixed” if we want.

Compartmentalization

The waterfall model of trust

Be aware of the direction of the trust flow.

For example:

OK: desktop ► system server ► compute node

OK: desktop ► system server ► login node

Compartmentalization

The waterfall model of trust

Be aware of the direction of the trust flow.

For example:

OK: desktop ► system server ► compute node

OK: desktop ► system server ► login node

Bad: desktop ► login node ► system server

Bad: login node console ► infrastructure server

Compartmentalization

The waterfall model of trust

Explain reasoning to all system staff.

- Yes, it may be inconvenient at times.
- Yes, it may save your cluster one day.

Enforce using account filters, firewalls, etc.

Compartmentalization

Separate servers

Users should only have access to the login nodes and, possibly, their allocated worker nodes.

Keep system servers separate from login nodes.

On large systems, separate system servers more if possible.

Example: File servers may require kernel versions with known local root exploits for weeks or forever. Restrict access to them!

Compartmentalization

Separate credentials

Do not use the same password or similar for different levels in the waterfall.

Do not use the same root password on different clusters.

Goal: Nothing you can steal at a lower level should gain you access at a higher level. But it's hard to fully reach this.

Compartmentalization

Separate credentials

Example:

Only staff can login to system servers. Good.

But staff homedirs are mounted on login nodes and compute nodes.

If you get root on them, you can become a staff member, change .profile, get to run code on system server (or boobytrap “ssh”, “su”, . . .)

Compartmentalization

Separate credentials

Be careful with single sign-on!

Example:

Library password

▼

LiU password

▼

Personal certificate

▼

Supercomputer access!

Hardening

There are a lot of things we can do. Some may conflict with ease-of-use, some not.

Hardening

suid stripping

Example: NSC antisuid runs from cron

- Whitelist of known binaries that are allowed to be setuid root – ping, sudo…
- Blacklist of known binaries that should have their suid bit stripped – mount…
- Any other suid binaries found? Sound the alarm!

Hardening

File system flags

Whenever possible, mount file systems:

- read-only
- nosuid, nodev
- root squash
- norelatime — performance vs. forensic abilities

Hardening

Module loading

Many kernel exploits depend on being able to load some obscure module. Questions to consider:

- Can we turn off module loading completely?
- Can we at least turn off autoloading of modules?

Hardening

Security frameworks

SELinux, grsecurity…

NSC doesn’t do this. Any success stories?

Introducing third-party dependencies may make updating harder.

Hardening

Help the users proactively

Examples of things you can do periodically:

- Check for really bad filesystem permissions
- Check for bad SSH usage
  - Unencrypted private keys
  - Known bad keys in authorized_keys (remember the Debian Debacle?)

And much, much more...

We still haven't covered things like log management, log analysis, intrusion detection...

And much, much more...

We still haven't covered things like log management, log analysis, intrusion detection...

...so here comes another 40 slides!

And much, much more...

We still haven't covered things like log management, log analysis, intrusion detection...

...just kidding!

And much, much more...

We still haven't covered things like log management, log analysis, intrusion detection...

...just kidding!

Questions? Opinions? Protests?