Securing Your Linux Servers

Thomas Cameron, RHCE, RHCI, CNE, MCSE, MCT
Lead Solutions Architect, Central Region
Red Hat

tcameron@redhat.com
http://people.redhat.com/tcameron
Outline

- Introduction
- Vectors of attack
  - External – network services
  - Internal – trusted and non-trusted users
- Installation time – trim your package list
- Pre-production – port scan, process scan & harden your server
  - Firewalls (iptables)
  - tcp_wrappers
Outline

- xinetd
- Security Enhanced Linux (SELinux)

Production

- Periodically port scan and check for unknown processes against baseline
Introduction

- Welcome!

- Today we will only be talking about hardening a single Linux host – 90 minutes is not enough time to go too terribly deep into security

- For security training, consider courses like:
  - RHS333, Red Hat Enterprise Security: Network Services
  - RHS429, Red Hat Enterprise SELinux Policy Administration

- For additional security information, see https://www.redhat.com/security/updates/
Vectors of Attack

- **External**
  - Most commonly considered attack vector.
  - Includes buffer overflows, dictionary/brute force attacks, malformed URLs and the like

- **Internal**
  - Attacks from within areas considered to be secure. More and more sensitive data is lost to internal attacks than external. Can be trusted staff who abuse privileges or non-trusted staff who exploit weaknesses of internal networks. Also consider social engineering - combination internal/external.
Installation Time

- Install only the minimum necessary software to do the job for which the system is designed.
  - A web server probably doesn't need gcc and friends installed.
- You can get a good handle on what software groups to install by looking at the various comps*.xml files
  - `egrep "<name>|packagereq" /path/to/comps.xml`
- Note that each channel will have its own comps file
  - Server, Cluster, ClusterStorage, VT
Installation Time

- Once you've determined the package list, generate a kickstart file.
  - Automated installs increase security by decreasing variance.
  - Automated installs decrease recovery time if compromised. If you know exactly how all your systems are configured, you know exactly what to fix on all of them.
Installation Time

- You can define the following in `%packages`:
  - Individual packages (package)
  - Package groups (@Group Name)
  - Packages to skip (-package)
Installation Time

- Use %post to:
  - Turn off unnecessary services (better you should not install any)
  - Fix settings like ipv6 or authentication
  - Add any user accounts (use /sbin/grub-md5-crypt to generate passwords)
  - Run any shell code needed to tighten things up
  - Install any packages outside the Red Hat install tree
Installation Time

install
url --url http://ml110.redhat.com/ty/gWwGX8y4
key 014fce2d4495830d
lang en_US.UTF-8

skip
network --device eth0 --bootproto dhcp
rootpw --iscrypted $1$agkorjqj$KeeFnItYAS.1.uUcjxrR1
firewall --disabled
authconfig --enablemd5 --enablesound

timezone America/Chicago
bootloader --location=mbr --driveorder=sda --append="rhgb quiet"

part /boot --fstype ext3 --size=100
part swap --size=1024
part / --fstype ext3 --size=1000 --grow

%packages
@base
@Web Server
-@tux

%post
/sbin/chkconfig foo off
/bin/rpm -Uvh http://ml110.redhat.com/pub/packages/mypackage-1.0-1.i386.rpm
/bin/echo install ipv6 /bin/true > /etc/modprobe.conf
/usr/sbin/groupadd admins
/usr/sbin/useradd -G admins -p '$1$uWyP9$RRM.DRsYpNcAUF6/KD/WV/' -c "Thomas Cameron" tcameron
Pre-production

Before your system goes into production:

- port scan
- process scan
- harden your server
Pre-production

- Port scan your server
  - nmap, nmapfe
  - Scan the host from another machine or machines
Pre-production

- nmap, nmapfe
  - Command line or GUI driven tools to do portscans
  - Don't portscan servers unless you know it is OK to do so!
    - Your systems
    - You have documented permission to scan them
Starting Nmap 4.11 (http://www.insecure.org/nmap/) at 2007-09-24 13:53 CDT
Interesting ports on vm1.redhat.com (172.31.103.128):
Not shown: 1677 closed ports
PORT STATE SERVICE
22/tcp open ssh
111/tcp open rpcbind
772/tcp open cycleserv2
MAC Address: 00:0C:29:36:4D:EB (VMware)
Device type: general purpose
Running: Linux 2.4.X|2.5.X|2.6.X
OS details: Linux 2.4.0 - 2.5.20, Linux 2.4.7 - 2.6.11
Nmap finished: 1 IP address (1 host up) scanned in 2.280 seconds

Command: nmap -sS -O -T1 -PT vm1.redhat.com
Pre-production

- Determine which services are supposed to be listening and make sure that those ports are open. If anything else is open, determine the service which is running and remove it.
  - `lsof -i :[port]` can help you determine which service is listening on what port
- Record what is listening – capture a baseline reading
```
[root@vml ~]# lsof -i :772
COMMAND      PID USER   FD   TYPE DEVICE SIZE NODE NAME
rpc.statd    1862 root   7u IPv4   5928 TCP *:cycleserv2 (LISTEN)
[root@vml ~]#  
```
Pre-production

- **Process Scanning**
  - Do a process scan to see what is running and who is running it.
    - `ps -aux`
    - `ps -ef`
Pre-production

- In the previous output, does it make sense to have hpiod running on, say, a web server?
<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th>hplip</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Version</strong></td>
<td>1.6.7</td>
</tr>
<tr>
<td><strong>Release</strong></td>
<td>4.1.el5</td>
</tr>
<tr>
<td><strong>Install Date</strong></td>
<td>Sat 22 Sep 2007 12:50:47 PM CDT</td>
</tr>
<tr>
<td><strong>Build Date</strong></td>
<td>Thu 16 Nov 2006 10:27:55</td>
</tr>
<tr>
<td><strong>Group</strong></td>
<td>System Environment/Daemons</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>12480519</td>
</tr>
<tr>
<td><strong>License</strong></td>
<td>GPL/MIT/BSD</td>
</tr>
<tr>
<td><strong>Signature</strong></td>
<td>DSA/SHA1, Wed 17 Jan 2007 09:36:50 AM CST, Key ID 5326810137017186</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hplip.sourceforge.net/">http://hplip.sourceforge.net/</a></td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td>HP Linux Imaging and Printing Project</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>The Hewlett-Packard Linux Imaging and Printing Project provides drivers for HP printers and multi-function peripherals.</td>
</tr>
</tbody>
</table>

[root@vm1 ~]#
Pre-production

- Probably not. Remove it or at least disable it.
  - `/sbin/chkconfig hplip off`
  - `/sbin/service hplip stop`
  - `yum remove hplip`

- Note that hplip is not necessarily a dangerous service to have running. It's just that it is not needed, and as such, should be removed.
Pre-production

- As with external ports, create a baseline of known acceptable running processes
- Don't rely on your memory. Document listening ports and running processes.
  - Easy for you to reference months or years later
  - Easy for other staff to reference should you not be available
Pre-production

Harden Your Systems

- iptables
- xinetd
- pluggable authentication modules (PAM)
- Security Enhanced Linux (SELinux)

Application Specific

- sendmail
- NFS
Harden Your Systems

- **iptables**
  - kernel-level stateful packet filter
  - Can be incredibly granular or very broad
  - `/usr/bin/system-config-securitylevel` great for basic host-based firewalls
  - `/sbin/iptables`, `/sbin/iptables-save`, and `/sbin/service iptables {start|stop|restart|condrestart|status|panic|save}` for more advanced configurations
Harden Your Systems

- `/usr/bin/system-config-securitylevel`
Harden Your Systems

- `/sbin/iptables`

```bash
[root@vml ~]# iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source          destination

Chain FORWARD (policy ACCEPT)
target prot opt source          destination

Chain OUTPUT (policy ACCEPT)
target prot opt source          destination
[root@vml ~]# iptables -I INPUT -s vm2 -p tcp --dport 22 -j ACCEPT
[root@vml ~]# iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source          destination
  ACCEPT tcp -- vm2.redhat.com anywhere tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target prot opt source          destination

Chain OUTPUT (policy ACCEPT)
target prot opt source          destination
[root@vml ~]# iptables -A INPUT -m state --state NEW -j REJECT
[root@vml ~]# exit
```
iptables (netfilter)

- Kernel-based traffic filter/firewall toolkit
- Traffic is managed through rules, called “chains:”
  - INPUT
  - FORWARD
  - OUTPUT
- Those packets can be sent to a target, including
  - ACCEPT
  - DROP
  - REJECT
iptables (netfilter)

- Additionally, packets can be modified using Network Address Translation
  - `-t nat`
    - DNAT (PREROUTING)
    - SNAT (POSTROUTING)
    - MASQUERADE (POSTROUTING)
iptables (netfilter)

- Some benefits of iptables:
  - Works at layers 2 and 3 – the application is never even aware that the traffic has hit the box. Lessens the chance of a buffer overflow or other malformed traffic attack.
  - Very low performance impact
iptables examples

- **First allow traffic to and from localhost:**
  ```
ipertables -A INPUT -i lo -j ACCEPT
  ```
- **Next allow icmp (ping and friends):**
  ```
ipertables -A INPUT -p icmp -j ACCEPT
  ```
- **To allow ssh from station101.example.com:**
  ```
ipertables -A INPUT -s station101 -p tcp \ 
  --dport 22 -j ACCEPT
  ```
iptables examples

- To allow traffic from the world to the web server:
  
  ```
  iptables -A INPUT -p tcp --dport http -j ACCEPT
  iptables -A INPUT -p tcp --dport https -j ACCEPT
  ```

- To allow all traffic from a trusted subnet:
  
  ```
  iptables -I INPUT -s 192.168.1.0/24 -j ACCEPT
  ```

- Now log any other traffic:
  
  ```
  iptables -A INPUT -m state --state NEW -j LOG \ 
  --log-prefix="Blocked Traffic "
  ```
iptables examples

- Finally, block any other traffic:
  iptables -A INPUT -m state --state NEW -j DROP
iptables examples

- Note that REJECT sends back an icmp port unreachable message, where DROP silently discards (and slows down the attacker)
iptables examples

- Don't forget to make your changes permanent!

```
[root@station100 ~]# iptables -L
Chain INPUT (policy ACCEPT)
  target  prot opt source            destination
  ACCEPT  all --  anywhere            anywhere
  ACCEPT  icmp --  anywhere            anywhere
  ACCEPT  all --  192.168.1.0/24      anywhere
  ACCEPT  tcp --  station101.example.com anywhere    tcp dpt:ssh
  ACCEPT  tcp --  anywhere            anywhere    tcp dpt:http
  ACCEPT  tcp --  anywhere            anywhere    tcp dpt:https
  LOG     all --  anywhere            anywhere    state NEW LOG level warning prefix 'Blocked Traffic'
  REJECT  all --  anywhere            anywhere    state NEW reject-with icmp-port-unreachable

Chain FORWARD (policy ACCEPT)
  target  prot opt source            destination

Chain OUTPUT (policy ACCEPT)
  target  prot opt source            destination

[root@station100 ~]# service iptables save
Saving firewall rules to /etc/sysconfig/iptables: [ OK ]
[root@station100 ~]#
```
tcp_wrappers

- Applications can be compiled against libwrap
- That application is going to check connections against /etc/hosts.allow first and then and /etc/hosts.deny
  - If a connection is allowed in hosts.allow then the connection will succeed. BE AWARE OF THIS!!!
- These files are configured in this format:
  <daemon list>: <client list> [: <option>: <option>: ...]
tcp_wrappers

- Where daemon is something like
  - vsftpd
  - sshd
  - in.telnetd

- And client list is something like:
  - 192.168.1. or 192.168.1.0/24
  - ALL
  - *.cracker.lan
tcp_wrappers

- And option is something like:
  - allow
  - deny
  - twist
  - spawn
  - except

- options can be added
  - sshd: 192.168.2. : spawn /usr/bin/logger “Access denied on $(/bin/date)”: deny
  - vsftpd : .example.com : twist /bin/echo "421 This domain has been black-listed. Access denied!"
tcp_wrappers

- Example of hosts.deny with a spawn directive to log:
tcp_wrappers

- Example of hosts.deny with a twist:
xinetd

- xinetd is the internet “super-server,” a replacement for the old inetd service.
  - A single service which listens on several ports. When a connection is made to a port, xinetd decides what service to start in response to that connection
  - Rules are stored in /etc/xinetd.d/[service]
xinetd

- Some security features for the service files
  - `only_from = 192.168.0.0/24`
  - `no_access = 192.168.1.10`
  - `access_times = 09:00-18:00`
xinetd

Logging options:

- **ATTEMPT** — Logs the fact that a failed attempt was made (log_on_failure).
- **DURATION** — Logs the length of time the service is used by a remote system (log_on_success).
- **EXIT** — Logs the exit status or termination signal of the service (log_on_success).
- **HOST** — Logs the remote host's IP address (log_on_failure and log_on_success).
- **PID** — Logs the process ID of the server receiving the request (log_on_success).
- **USERID** — Logs the remote user using the method defined in RFC 1413 for all multi-threaded stream services (log_on_failure and log_on_success).
xinetd

- Example configuration file for the kerberos-enabled telnet service, /etc/xinetd.d/krb5-telnet:

```
service telnet
{
    flags = REUSE
    socket_type = stream
    wait = no
    user = root
    server = /usr/kerberos/sbin/telnetd
    log_on_failure += USERID
    disable = yes
}
```
xinetd

- Using the SENSOR attribute with xinetd-managed services
  - SENSOR is a sort of “honeypot” tool
    - Once an unauthorized system attempts to connect, xinetd locks the entire system down for that system
xinetd

Using SENSOR:

```plaintext
service telnet
{
  flags = SENSOR
  deny_time = 1
  socket_type = stream
  wait = no
  user = root
  server = /usr/kerberos/sbin/telnetd
  log_on_failure += USERID
  disable = no
}
```
xinetd

- Using SENSOR:

  Sep 25 01:13:50 station100 xinetd[2437]: START: login pid=2442 from=192.168.1.101
  Sep 25 01:13:54 station100 xinetd[2437]: EXIT: login status=0 pid=2442 duration=4(sec)
  Sep 25 01:13:58 station100 xinetd[2437]: 2437 {process_sensor} Adding 192.168.1.101 to the global_no_access list for 1 minutes
  Sep 25 01:13:58 station100 xinetd[2437]: FAIL: telnet address from=192.168.1.101
  Sep 25 01:14:02 station100 xinetd[2471]: FAIL: login address from=192.168.1.101

  Sep 25 01:14:58 station100 xinetd[2437]: At least 1 DENY_TIME has expired, global_no_access list updated
  Sep 25 01:14:58 station100 xinetd[2437]: global_no_access list is empty.
Another important feature of xinetd is its ability to control the amount of resources which services under its control can utilize.

It does this by way of the following directives:

- `cps = <number_of_connections> <wait_period>` — Dictates the connections allowed to the service per second. This directive accepts only integer values.
- `instances = <number_of_connections>` — Dictates the total number of connections allowed to a service. This directive accepts either an integer value or UNLIMITED.
xinetd

- It does this by way of the following directives (Continued):
  - `per_source = <number_of_connections>` — Dictates the connections allowed to a service by each host. This directive accepts either an integer value or UNLIMITED.
  - `rlimit_as = <number[K|M]>` — Dictates the amount of memory address space the service can occupy in kilobytes or megabytes. This directive accepts either an integer value or UNLIMITED.
xinetd

- It does this by way of the following directives (*Continued*):
  - `rlimit_cpu = <number_of_seconds>` — Dictates the amount of time in seconds that a service may occupy the CPU. This directive accepts either an integer value or UNLIMITED.
Security Enhanced Linux (SELinux)

- Security-Enhanced Linux (SELinux) is a security architecture integrated into the 2.6.x kernel using the Linux Security Modules (LSM). It is a project of the United States National Security Agency (NSA) and the SELinux community. SELinux integration into Red Hat Enterprise Linux was a joint effort between the NSA and Red Hat.
Security Enhanced Linux (SELinux)

- SELinux provides a flexible Mandatory Access Control (MAC) system built into the Linux kernel. Under standard Linux Discretionary Access Control (DAC), an application or process running as a user (UID or SUID) has the user's permissions to objects such as files, sockets, and other processes. Running a MAC kernel protects the system from malicious or flawed applications that can damage or destroy the system.
Security Enhanced Linux (SELinux)

Refer to the following diagram:
Security Enhanced Linux (SELinux)

- SELinux Operating Modes

  Instead of running in enforcing mode, SELinux can run in permissive mode, where the AVC is checked and denials are logged, but SELinux does not enforce the policy. This can be useful for troubleshooting and for developing or fine-tuning SELinux policy.
Security Enhanced Linux (SELinux)

- Files Related to SELinux
  - The `/selinux/` pseudo-file system contains commands that are most commonly used by the kernel subsystem. This type of file system is similar to the `/proc/` pseudo-file system.
  - Administrators and users do not normally need to manipulate this component.
Files Related to SELinux (cont'd)

The following example shows sample contents of the `/selinux/` directory:

- `rw-rw-rw- 1 root root 0 Sep 22 13:14 access`
- `dr-xr-xr-x 1 root root 0 Sep 22 13:14 booleans`
- `--w------- 1 root root 0 Sep 22 13:14 commit_pending bools`
- `rw-rw-rw- 1 root root 0 Sep 22 13:14 context`
- `rw-rw-rw- 1 root root 0 Sep 22 13:14 create`
- `--w------- 1 root root 0 Sep 22 13:14 disable`
- `rw-r--r-- 1 root root 0 Sep 22 13:14 enforce`
- `rw------- 1 root root 0 Sep 22 13:14 load`
- `r--r--r-- 1 root root 0 Sep 22 13:14 mls`
- `r--r--r-- 1 root root 0 Sep 22 13:14 policyvers`
- `rw-rw-rw- 1 root root 0 Sep 22 13:14 relabel`
- `rw-rw-rw- 1 root root 0 Sep 22 13:14 user`

For example, running the `cat` command on the `enforce` file reveals either a 1 for enforcing mode or 0 for permissive mode.
SELinux Configuration Files

- The following sections describe SELinux configuration and policy files, and related file systems located in the /etc/ directory.
Security Enhanced Linux (SELinux)

- **SELinux Configuration Files**

  - The `/etc/sysconfig/selinux` Configuration File
    - There are two ways to configure SELinux under Red Hat Enterprise Linux: using the Security Level Configuration Tool (`system-config-selinux`), or manually editing the configuration file (`/etc/sysconfig/selinux`).
### SELinux Configuration Files

- The `/etc/sysconfig/selinux` file is the primary configuration file for enabling or disabling SELinux, as well as for setting which policy to enforce on the system and how to enforce it.

**Note:**
- The `/etc/sysconfig/selinux` is a symbolic link to the actual configuration file, `/etc/selinux/config`. 
Security Enhanced Linux (SELinux)

- SELinux Configuration Files
  - The following explains the full subset of options available for configuration:
  - SELINUX=enforcing|permissive|disabled — Defines the top-level state of SELinux on a system.
    - enforcing — The SELinux security policy is enforced.
SELinux Configuration Files

- permissive — The SELinux system prints warnings but does not enforce policy.

  ✓ This is useful for debugging and troubleshooting purposes. In permissive mode, more denials are logged because subjects can continue with actions that would otherwise be denied in enforcing mode. For example, traversing a directory tree in permissive mode produces avc: denied messages for every directory level read. In enforcing mode, SELinux would have stopped the initial traversal and kept further denial messages from occurring.
Security Enhanced Linux (SELinux)

- SELinux Configuration Files
  - disabled — SELinux is fully disabled. SELinux hooks are disengaged from the kernel and the pseudo-file system is unregistered.
Tip

Actions made while SELinux is disabled may result in the file system no longer having the correct security context. That is, the security context defined by the policy. The best way to relabel the file system is to create the flag file /.autorelabel and reboot the machine. This causes the relabel to occur very early in the boot process, before any processes are running on the system. Using this procedure means that processes can not accidentally create files in the wrong context or start up in the wrong context. It is possible to use the fixfiles relabel command prior to enabling SELinux to relabel the file system. This method is not recommended, however, because after it is complete, it is still possible to have processes potentially running on the system in the wrong context. These processes could create files that would also be in the wrong context.
Security Enhanced Linux (SELinux)

- SELINUXTYPE=targeted|strict — Specifies which policy SELinux should enforce.
  - targeted — Only targeted network daemons are protected.
  - strict — covers everything (use with caution only if you really understand what you are doing)
Security Enhanced Linux (SELinux)

Important

- The following daemons are protected in the default targeted policy: dhcpd, httpd (apache.te), named, nscd, ntpd, portmap, snmpd, squid, and syslogd. The rest of the system runs in the unconfined_t domain. This domain allows subjects and objects with that security context to operate using standard Linux security. The policy files for these daemons are located in /etc/selinux/targeted/src/policy/domains/program. These files are subject to change as newer versions of Red Hat Enterprise Linux are released.
Security Enhanced Linux (SELinux)

- Policy enforcement for these daemons can be turned on or off, using Boolean values controlled by the Security Level Configuration Tool (system-config-selinux).
- Setting a Boolean value for a targeted daemon to 0 (zero) disables policy transition for the daemon. For example, you can set dhcpd_disable_trans to 0 to prevent init from transitioning dhcpd from the unconfined_t domain to the domain specified in dhcpd.te.
Security Enhanced Linux (SELinux)

- Use the `getsebool -a` command to list all SELinux booleans. The following is an example of using the `setsebool` command to set an SELinux boolean. The `-P` option makes the change permanent. Without this option, the boolean would be reset to 1 at reboot.

  ```bash
  setsebool -P dhcpd_disable_trans=0
  ```

- **strict** — Full SELinux protection, for all daemons. Security contexts are defined for all subjects and objects, and every action is processed by the policy enforcement server.
SELinux Utilities

• `/usr/sbin/setenforce` — Modifies in real-time the mode in which SELinux runs.
  
  • For example:
    ✓ `setenforce 1` — SELinux runs in enforcing mode.
    ✓ `setenforce 0` — SELinux runs in permissive mode.
  
  • To actually disable SELinux, you need to either specify the appropriate `setenforce` parameter in `/etc/sysconfig/selinux` or pass the parameter `selinux=0` to the kernel, either in `/etc/grub.conf` or at boot time.
SELinux Utilities

/usr/sbin/sestatus -v — Displays the detailed status of a system running SELinux. The following example shows an excerpt of sestatus -v output:

SELinux status: enabled
SELinuxfs mount: /selinux
Current mode: enforcing
Mode from config file: enforcing
Policy version: 21
Policy from config file: targeted

Process contexts:
Current context: user_u:system_r:unconfined_t:s0
Init context: system_u:system_r:init_t:s0
/sbin/mingetty system_u:system_r:system_t:s0
Security Enhanced Linux (SELinux)

- SELinux Utilities

  - `sbin/restorecon` — Sets the security context of one or more files by marking the extended attributes with the appropriate file or security context.

  - `sbin/fixfiles` — Checks or corrects the security context database on the file system.

  - `usr/bin/chcon` — changes the security context on the filesystem
Security Enhanced Linux (SELinux)

- SELinux Examples
  - If an application is being blocked, the first thing you can do is use restorecon -vR against the directory.
  - If you have installed an app which does not have an associated policy, you can use chcon to set the context of the application.
Security Enhanced Linux (SELinux)

- Example – you've downloaded the newest version of Firefox and installed it in /usr/local
  - `restorecon -vR /usr/local`
Security Enhanced Linux (SELinux)

- Example – to demonstrate how MAC can override DAC

  - A bad guy gets control of an application like Apache and instructs Apache to read a sensitive file like /etc/shadow

  - `chcon --reference /etc/shadow /home/moe/public_html/secret.txt`
Security Enhanced Linux (SELinux)

- Example – to demonstrate how booleans can be used to turn on or off access
  - To turn off home directory access:
  - `setsebool httpd_enable_homedirs off`
Production

- Once the server is in place, use utilities such as aide to check system integrity.
  - `rpm -Va` is another useful utility, but since the RPM database is stored on the local filesystem, it is not reliable
- Periodically check for any open ports or processes which were not there during the build and hardening process
- Set up remote logging - see the `/etc/sysconfig/syslog` file for details, it's very well documented.
- Use logwatch to monitor your logs, it's included and active by default