Co-funded by the Erasmus+ Programme of the European Union

ISSES – Information Security Services Education in Serbia

Supported by the Erasmus+ Capacity Building in the field of Higher Education (CBHE) grant N° 586474-EPP-1-2017-1-RS-EPPKA2-CBHE-JP

OS HARDENING

COMPUTER SECURITY

Lecture 7

ISSES 2017-2020
Novi Sad, Serbia

Information Security Services Education in Serbia (ISSES)

7.1 CONCEPT OF HARDENING

What is hardening?

- A general purpose OS, when installed, tends to prioritize *flexibility* and *functionality.*
- You install an OS for a purpose: you want your computer to do something.
- Security, by its very nature, never allows for things, only disallows them.
- Therefore, after first install, general purpose OS installs tend to be very low on security.
- They permit as much as possible so as to let the user accomplish as much as possible.
- The idea behind this approach is to permit much, configure for desired functionality, and then tighten security.

Security first?

- Needless to say this is psychologically quite risky.
- Once you have the OS doing what you want it to, it's tempting to leave it at that, leaving essential security for 'later.'
- Bitter experience shows that this 'later' tends to never happen.
- However, there are technological reasons for this.
- Security is an impedance by its very nature
- When configuring a system that has been tightened down, you face two unknowns: does something not work because it is not configured correctly, or does it not work because *security* is not configured correctly, and is blocking a legitimate function?

Hardening

- Therefore the practice remains for the default install to be open (or relatively open)
- Once all desired functionality has been configured the system is *hardened*
- This process enhances security through two principal methods
  - Additive
  - Substractive
- Additive enhancements add new security mechanisms, catching problems or providing additional checks and protections.
- Substractive enhancements remove any feature of the system that's not needed.

Hardening

- One can view additive measures as toughening the attack surface, making it more difficult to penetrate.
- Similarly, substractive measures make the attack surface itself smaller.
- An example of the former is protecting some vital security function with a password. The functionality is still here, we just added an additional layer of security around it.
- An example of the latter, is removing a mail server from a machine not intended to act as one. There's no need for this facility, therefore having it without it being needed increases risk for no commersuate reward.

Rules and authorities

- Hardening an OS is not something done trivially
- Certainly, as you improve as an expert, you can intuit what should be done more andd more, but in the end to mount an effective defense against a determined attacker you need to get *everything* right.
- An attacker just needs to get *something* right.
- As a result, you should never harden an OS 'by ear.'
- The system should be hardened according to an established standard.
- The rules are much like encryption *never roll your own* and *if you don't know what it does, assume it is critical.*

Sources

- This lecture is heavily based on the NIST-certified, open source Open SCAP project which we will be using extensively during your tutorials. https://www.open-scap.org
- During our discussion on hardening practices we will often cite specific rules (typically from the RHEL7 guide, in particular) in the form of, e.g. CIS CCE-27309-4, which indicates a that the Commercial Cloud Services baseline based on CIS v2.1.1. is used, and that of those a specific rule is invoked.
- The structure of the lecture is inspired, at least in part, by the Practical Linux Hardening Guide.

Practical instructions

- You will note that this lecture does not include detailed step-by-step instructions on how to do all the things it suggests should be done.
- This is partially because those instructions depend on the specific distribution being used while this is meant to more outline a basic set of hardening rules.
- However, during your lab periods you will go through a practical hardening exercise based on this lecture.
- In those lab periods, you will see the precise instructions to take, while in this lecture you will learn what those steps *mean.*

Information Security Services Education in Serbia (ISSES)

7.2 HARDENING BASIC INFRASTRUCTURE

Boot loader hardening

- We have mentioned that physical access is a nightmare, security-wise.
- Even if we are reasonably sure that the actual hardware is safe behind locks and electronic tripwires, just access to a console may be enough to break into our machine with suprising ease.
- How?
- Operating systems don't start themselves, generally some sort of loader software is necessary to get them started.
- The standard for Linux is GRUB.
- Seeing as not being able to get into your system due to a misconfiguration is a serious problem GRUB has options for rescuing your installation.

Boot loader hardening

- This is why, when you start a computer with GRUB you can cause it to pause at its menu.
- Then with the 'e' option you can to edit the command line needed to start the OS. The command option to start the OS can be quite complicated.
- However, the important part is the command starting with 'linux' which specifies the path to the kernel, the device where the kernel is (these days specified with an UUID) and similar initial options.
- Just adding init=/bin/bash is all you need to do.
- Press F10 and you will boot into single user mode.
- This means you will skip most of the startup and be dropped into a root shell. You win.

Boot loader hardening

- Now, strictly speaking, you can't really do much here because the root partition is mounted read-only. This is the equivalent of a rescue prompt: only what's needed for a minimal system rescue.
- But with a mount –n –o remount, rw / command, you can remount the root partition in read-write mode and with that you can do anything you want to the system. This includes reseting anyone's password, creating new, hidden sysadmin accounts, or installing a root-kit on the machine.
- This sounds *terrible* but there's a good reason for it: if you are debugging a faulty installation this may be the only way to get it to work How to use it to make sure anyone passing by can completely control the machine?

Boot loader hardening

- Accoding to CIS CCE-27309-4 the solution is to generate a password for the bootloader.
- The system still boots fine, but if the user wishes to change anything, then a password is required.
- The process requires the grub2-setpasword comand, editing the grub users file and regenerating the configuration.
- Of course this is a production-use non-personal password, meaning that it will have to comply with a suitable password generation and retention policy.

Bootloader hardening

- Since we went to such trouble to configure the bootloader to have a password it's important to not let anyone but the root user have access to boot loader configuration.
- Thus, /boot/grub2/grub.cfg, /etc/grub/conf, and /etc/grub.d should all have permission level 600 and belong to root.
- CIS CCE-27054-6
- In most cases something similar to this is already in force by default, but for instance in Ubuntu 20.10 the default config reserves writing privilege for the root user, but permits anyone to read.
- There is no reason to permit access to even a hashed password, let alone system configuration files to any user. This is best remedied.

Secure partitioning

- CCE-80144-9, CCE-26404-4, CCE-27173-4, CCE-26971-2, and CCE-26967-0 recommend careful partitioning of the disk suggesting that the following should be mounted on separate partitions:
  - /home
  - /var/tmp
  - /var
  - /tmp
  - /var/log/audit
  - /var/log
- Other paritions may be relevant in case they are used. Typically /var/www is mounted separately too if the server is used for web hosting.

Secure partitioning

- Why partition the disk? What does that have to do with security?
- The security impact is not great but it is still a useful thing
- Careful partitioning helps prevent a runaway process (possibly affected by a malicious agent as a part of a denial of service attack) from filling up the entire hard drive.
- Running out of hard drive space is bad, and frequently software does not know how to handle not being able to create files, especially temporary files, gracefully.
- This makes exhausting disk space an effective way to effect a DOS attac.
- Clever partitioning limits the damage and reduces the attack surface.

Secure mount options

- An additional benefit to keeping things on separate partitions is that, when mounting a partition, certain broad options can be added that affect all files on that partition.
- This can help insulate an entire section of a system from certain malicious actions.
- Of particular interest are the nosuid and nodev options (which can be added in the /etc/fstab file).
- Nosuid is the simpler one and blocks the existance of setuid files.
- Setuid files, unless used *very* carefully, are a security liability and removing them entirely from a partition where we don't expect to see them at all reduces the attack surface at little cost.

Secure mount options

- Nodev is similarly 'cheap' and specifies that the given mounted partition cannot create device files.
- While normally the system generates all the needed dev files automatically in /dev, there's normally nothing stopping a user making a character device file with mknod, say, in our own home directory.
- Nodev removes the possibility and removes another attacks surface in a way which, for a vast majority systems, is basically free.
- In normal operation this isn't something you want to do anyway, so it is an easy improvement.

Noexec and going too far

- Another incredibly powerful mount option is 'noexec.'
- Noexec simply declares that an entire partition can't have any executable files on it, at all.
- Obviously, some destinations like /var can be safely made noexec for no binaries are meant to be there in the first place.
- But noexec is too powerful to use indiscriminately
- If you have users, they may find it a given that they can make their own executables in their own home directory and a policy which forbids all of that is unlikely to be popular.

Selective use of ro

- Ro is a mount option which specifies that a partition is to be mounted read-only.
- This can be changed (by remounting) but that requiers a very great deal of privilege on the system.
- A read-only mounted partition only makes sense for something which changes infrequently enough that it is worth your while to remount it read-write for that change.
- A great example of a use-case for this sort of protection is the /boot partition where the linux kernel images and necessary configuration needed to run the OS are kept.
- This is something we change rarely (only when upgrading the kernel) and which most of the time *shouldn't* be changed.

Limiting /dev/shm

- Modern systems typcially have mounted a tmpfs instance on /dev/shm
- This permits a temporary file system which is memory-backed thus allowing for incredibly rapid and effective temporary file creation.
- However, this opens a DOS attack surface as someone can exhaust RAM by creating too many files.
- Luckily, the size parameter, in this instance, can be used to limit the size of such a system to something well under the total amount of system present in the memory.

Proc protection

- System-internal information must be guarded carefully on secure systems.
- This includes normally harmless stuff
- Maybe something may not harm anything, but unless we are very certain and we do not need acess to it, it's always better to remove that access as soon as possible.
- An example of this is the /proc file system which normally permits users to see details about all processes.
- To remove this simply add a 'hidepid' option in the mount settings and then set it to level 2.
- This will hide all processes that are not our own from the /proc directory structure.

Unexpected encryption

- The notion of encrypting normal partitions is relatively normal.
- Modern CPUs have hardware-based AES encryption which means that the CPU overhead is low and the access speeds are high.
- However, it is not uncommon to also encrypt swap partitions.
- Why?
- Swap can contain *any* memory. This includes, say, decrypted secret keys, confidential information, banking details...
- It's best that it is never made easily available

Information Security Services Education in Serbia (ISSES)

7.3 Software sources and removing uneeded services

Securing software sources

- No amount of hardening will help if the software installed on the system is in itself compromised.
- This is why updates are very important. Of course, you don't need a hardening guide to tell you to update your software.
- A more important step is to make sure that software installation and update works in a secure fashion.
- CIS CCE-26989-4 in particular recommends engaging automated cryptographic signature checking for all packages.
- Likewise, the keyring used for the package manager should be inspected so that it only trusts the right keys.

Securing software sources

- If possible, package managers should be configured to also use HTTPS.
- Strictly speaking this is not needed, since signatures are verified already, but it provides an additional layer of security.
- Needless to say, introducing a malicious repo is a very good way to compromise the system, thus source lists should be rigorously controlled.

Purging dangerous software

- It's very common for linux installs to contain considerably more tools and applications than needed.
- In particular, very frequently legacy tools are installed in the interests of compatiblity and because they use very little system resources.
- The notion is that the end-user will either use them or not, but if they aren't used they won't do any harm.
- However, each such app increases the attack surface.
- Each thing the computer does is another place to attack and one more place to secure and keep continuously updated.
- Therefore, anything not needed should be disabled and this goes in particular for apps which are notorious for their bugs.

Legacy remote administration

- Systems often still ship with tools for remote administration and control of systems that are obsolete and have poor security.
- Particularly problematic is the 'r' suite of tools: rsh, rlogin, and rexec.
- These are tools for remote system control and command execution which are obsolete on account of lacking any encryption.
- This means that all login details are sent in cleartext which means that a malicious actor on the same network could launch a man-in-the-middle attack.
- SSH does everything these things do, better and safer. There's no reason they should stay.

Legacy remote administration

- It is also important to remove the hosts.equiv and .rhosts files which allow loginless use from given hosts.
- This is exceptionally unsafe and those files should be removed if for some reason R-services are still present.
- Much like the R-services, telnet is a remote administration protocol with no guaranteed encryption. There is no reason for a telnet port to be open.
- These are generally not functional on modern systems already, but it's still something that needs to be checked.
- A number of less secure IoT solutions have open telnet ports.

Legacy authentication services

- The NIS protocol (originally named Yellow Pages, but renamed due to trademark issues) is used to share information regarding users between systems.
- It occupied a technological niche today occupied by LDAP and Kerberos.
- However, unlike these, the traffic was not encrypted and the authentication mechanism was not as effective as the sophisticated system of Kerberos.
- Therefore, disabling NIS through removing the 'ypserv' from the system is highly recommended.

Xinetd

- Xinetd is a metaserver used to provide united network listening functionality and wrapping functionality for multiple services.
- It is no longer necessary for most things so it can, generally speaking, be removed.
- Removing it is a good idea as an attack can, therefore, go through xinetd *or* the server it is listening for doubling our attack surface.
- Xinetd had one of the worst vulnerabilities of all time, CVE-2001-0825 which permitted arbitrary code execution to anyone with no special requirements whatsoever, leading to a CVSS score of an even 10.0.

Removing (T)FTP support

- TFTP is the trivial file transfer protocol, a deliberately as-simple-as-possible file transfer mechaism which does not require authentication and does not perform encryption.
- What it gains in simplicity it loses in security, of which it has none.
- It's highly unlikely that a production server has any need of it, threfore it should be disabled and, possibly, even removed entirely from the computer.
- Similarly, any normal FTP servers should still be removed. FTP adds authentication, but does not provide encryption, and it can be assumed that very nearly any network protocol whatsoever that does not employ TLS is a security risk that's best removed.

X windows

- X windowing system a.k.a. X11, and its inheritors such as Xorg and Wayland have a lengthy history of security vulnerabilities.
- Since they are so fundamental the latest versions are regularly patched and kept secure but the risk remains, given that they are incredibly complex pieces of software.
- There is no reason for a production server machine to have X, so for computers in such an environment it is recommended in the strongest possible terms to remove X windows entirely.

Avahi

- Avahi is, unlike most things listed here, not obsolete nor unneded in modern operations.
- Most modern distributions have it and it really doesn't have a secure alternative.
- It's a deamon enabling service advertising and discovery on local networks through an implementation of the mDNS/DNS-SD zerofconf protocol suite.
- It's the software that tells your laptop when you power it on within your home WiFi that your printer's ready to accept printing or that your NAS is sharing files without you having to enable this functionality on purpose.
- In an office or school environment it is a vital piece of software.

Avahi

- Despite this CIS CCE-80338-7 suggest it be disabled.
- Why? It leaves open a network port which provides an avenue of attack.
- This should be removed unless it is running on a perfectly trusted local network and the service is required.
- For servers the service is typically *not* required and reperesent an unecessary risk.
- Therefore the hardening policy is to remove avahi if not needed for system functionality.
- In general even quite secure software *if not needed for system operation should be removed.* For instance CIS CCE-80325-4 states that OpenLDAP should be removed: not because OpenLDAP is insecure but because, if not needed, it should not be running.

Information Security Services Education in Serbia (ISSES)

7.4 BASIC SECURITY INFRASTRUCTURE

Taking time into account

- Any secure system should keep accurate time for time-sensitive auth tasks such as kerberos, keeping accurate logs, and providing recurring security checks and services.
- Therefore either chronyd or ntp (the network time protocol) services should be running to ensure that the system time is always accurate.
- Likewise, crond should be enabled in order to make sure scheduled tasks can be executed.
- Scheduled tasks can provide an automated mechanism for intrusion detection, advanced logging, self-checks, and keeping software and configurations up to date.

Tcp_wrappers

- CIS CCE-27323-5 suggests intalling and enabling tcp_wrappers.
- This is a pacakge that provides a library that software can be linked against and configuration files hosts.allow and hosts.deny which exist in the /etc system directory.
- Using allow, IPs can be explicitly whitelisted for use.
- Using deny, IPs can be explicitly blacklisted for use.
- The allow list is read ahead of the deny list.
- This means for private server, we can deny everything, and then explicitly allow only trusted hosts that are meant to use our services.
- For a public server, however, we have to use a more limited form of denying only specific known-bad hosts that are, e.g. the sources of persistent attacks.

Issues with tcp_wrappers

- While tcp_wrappers provide a valuable service and they have not had their security compromised, their long-term viability is in question.
- The codebase is old: over twenty years old, in fact, and it has no open source community around it.
- This makes it a long-term liability in case vulnerabilities are found or security praxis should change.
- Fedora 28, for instance, has them deprecated, though it is unlikely any distribution will take steps to remove it.
- What's the alternative, then? Generally these days firewalls are best used to provide tcp_wrapper-like functionality with the added benefit of being more configurable at the cost of requiring more work to configure adequately.

Set up accounts for greater security

- We already stopped a major way for booting the machine in single-user mode through a GRUB password, but this is not foolproof. Issues in the boot process (which a malicious agent may cause) can lead the process to dump the user into single-user (and therefore root) mode.
- Therefore, it is best to configure the rescue and emergency services to demand authentication for single-user mode. (CIS CCE-27287-2).
- CIS CCE-27175-9 demands the removal of all UID 0 accounts save root in order to reduce the target surface.
- Lastly, the root user should be configured in such a way as to disable direct logins (CIS CCE-27294-8). The root user should only be accessible through an intermediate administration account.

Login banner

- It is a part of hardening to set up a login banner for normal and SSH logins warning anyone logging in about their rights and obligations.
- A typical such banner is offered by Cisco through their Baseline Security document:

```
UNAUTHORIZED ACCESS TO THIS DEVICE IS PROHIBITED
 You must have explicit, authorized permission to access or configure this device.
 Unauthorized attempts and actions to access or use this system may result in civil and/or
criminal penalties.
 All activities performed on this device are logged and monitored.
```

- The purpose of such a warning isn't to scare anyone off, but to instead to provide a legal backing: this means that an attacker can't claim they weren't aware the system wasn't open, and even more importantly, warns people regarding expectation of privacy.

Login banner

- The preceeding does not constitute legal advice, however.
- The precise form of the banner depends heavily on the cyber-security and privacy laws of the country in which jurisdiction any incident would be investigated and prosecuted in.
- If hardening an OS for academic purposes, this is not important.
- If hardening an OS for an institution the banner text should be drafted by or at the very least approved by the legal council of said institution.
- Your job, as a security professional is to ensure that there is such a banner, not what it says.

Sensible session defaults

- On a unix system it is possible to control the defaults that apply to user settings on the system.
- This is done through the /etc/profile and /etc/bashrc files
- One of those settings is the umask which determines the default permissions of new files creted by the user.
- The typical value in a lot of distributions (e.g. Ubuntu 20.10) set a default umask value of 022 for all users.
- This means any file you don't explicitly protect is world-readable.
- An umask of 027 is recommended for normal users and an umask of 077 for the root user.

Password policy

- When configuring accounts, one must also set the password policy that the owners of those accounts must adhere to.
- Users tend to want to use simple, easy to remember passwords which they never change and reuse everywhere.
- It is the job of the password policy to frustrate the users in all these desires.
- The two methods of doing this are password form policies and password aging polcies.
- The former control what a password looks like and are meant to stop easy to guess passwords and the latter are meant to stop password reuse and force the user to change passwords regularly.

Password policy

- When it comes to passwords the key thing is to make passwords hard to guess mechanically.
- Length is the best way to do this.
- It's common to demand numerals and special characters, under the assumption that as the attacker is brute-forcing the password they'll be forced to guess all possible combinations of all characters and expanding the character set is a cheap way to increase the number of guesses.
- This is true but the harder a password is for a *person* to remember, the higher the probability of irresponsible password use policies.
- Therefore, for a regular user, it may be wise to focus on a lengthy easy-to-remember password, rather than one with complex characters.

Password policy

- A password consisting of 4 randomly chosen all lowercase words from a dictionary with 100 000 words is relatively easy to remember, and offers approximately 66 bits of security even in the worst possible case of the attacker knowing the form of the password and the dictionary being used.
- Passwords should also be set to age, demanding that the user change them after 60-90 days, depending on the security policy being followed (CIS says 90).
- Passwords should also have a minimum age (CIS says 7 days).
- Why this? Because otherwise users can set a new password, satisfying the system, and then change it back to the old one. The system can't detect reuse then.

Verify basic permission integrity

- A basic move in hardening a system is to verify that all security-critical files have the correct ownership and permissions.
- Of particular importance, for obvious reasons are /etc/passwd /etc/group /etc/shadow and /etc/gshadow
- Further, the system should be searched for both malformed and unsecured files.
- Malformed files belong (through an error or an incompletely deleted user) to no user or no group. The find command offers –nouser and –nogroup switches to simplify finding this.
- Unsecured files, on the other hand, are world-writable. This should be corrected.

Permissions of core dump files

- A core dump is an image of process memory created at the instance of a crash. The name comes from the now-obsolete technical jargon of 'core' meaning 'memory.' It refers to the long-disused ferrite core memory common from mid fifties up to the seventies.
- Such a dump may contain system-critical data, and studying it may give an attacker vital information.
- Even if the dump has no crucial data it still gives valuable insight into the internal workings of the crashed process. This makes it easier to launch memory-based attacks on it.
- It is common to disable core dumps completely by using the pam.limits module to set the hard limit on core size to zero.
- The alternative is to limit acess to such files to only certain users.

Configure PAM

- PAM lets sysadmins control how users are authneticated, often at an application level.
- The first thing to configure as part of PAM is the password hash algorthim.
- Current best practice is to use a SHA-512 based hash (sha512crypt) which uses a NIST-approved SHA-2 standard with 512 bits. Given that this type of hash undergirds Bitcoin, as long as that has not collapsed, you can be sure that your system is safe from that front.
- This is what's used by default but it should be at least verified.
- Each password hash in /etc/shadow is marked by the code of the hashing scheme used, allowing you to be sure.

PAM password policies

- PAM allows us to lock out users after too many failed password attempts.
- Typical values are locking out users after five failed attempts and having the account automatically unlock after 900 seconds.
- This can be tweaked depending on the desired level of security.
- Likewise, PAM can be configured, via the remember command, to keep a log of previous user passwords (hashed, naturally) so that any attempt to reuse a password is detected and prevented.
- With password minimum age policies we mentioned already, any password can only be reused after 35 days have ellapsed.

OpenSSH hardening

- SSH is *the* solution for remote control and file transfer.
- It's fast, powerful and encrypted.
- However, that naturally means that any misconfiguration is very dangerous.
- As a result it's important to configure OpenSSH correctly.
- OpenSSH should be configured to never permit a login without a password. Our password policy ought to make that impossible anyway, but it is good to be sure.
- Likewise, there should be an idle timeout for connections: a connection that's not over and has been terminated from the client side can possibly be hijacked.

OpenSSH hardening

- SSH should be configured to use the second version of the protocol exclusively as SSHv1 is not safe.
- SSH should be configured to not permit users to configure their own SSH environment options: that might allow users to accidentally lower the oversal security of the system.
- SSH should be set to not trust .rhosts or allow host-based authentication: trusted hosts mean that one compromise expands near-instantly to everything being compromised.
- SSH should be set to the highest level of logging, the INFO level so that the logs are maximally clear.
- SSH should either ban X11 connections over SSH or, if they are needed, ensure they are encrypted through the X11Forwarding parameter

OpenSSH auth hardening

- OpenSSH support an array for ciphersuites for authentication and encryption.
- It is possible to disable some and enable others, in this case only modes which are widely thought to be secure should be left enabled. According to CIS CCE-27295, this list is: aes128 in CTR and CBC modes, aes192 in CTR and CBC modes, aes256 in CTR and CBC modes, and 2des in CBC mode. This is the minimum. If you have control over the clients you can be even more secure and only permit modes of aes256.
- MaxAuthTries should be set in order to minimize the chance of brute-forcing. While it is possible to detect iterated attacks as they happen and act, this provides a fail-safe in case the attempt is not detected.

OpenSSH auth hardening

- The PermitRootLogin option should be set to 'no.'
- Root login should not be permitted at all, let alone over a network connection.

Information Security Services Education in Serbia (ISSES)

7.5 AUDIT CONTROLS

Auditd

- Auditd is a mechanism for recording system activities for purposes of logging
- It's not the only auditing solution and you will learn a great deal more reagarding auditing, logging, and the analysis of the results of such systems in other subjects.
- For us, here and now, auditd is a fine enough example.
- The first thing to do is to get auditd to start early.
- All processes have a tag which indicates if they are audited or not, and auditd sets that tag for all processes that start after it starts running.
- Starting the auditd process earlier (as through the audit=1 option in the grub command line) will make sure it's present throghout the boot process. CIS CCE-27212-0.

Auditd log management

- Audtid should be configured to be careful about its logs in the /etc/audit/auditd.conf file.
- You should configure
  - the maximum total log file size,
  - notifications for low disk space,
  - log rotation when one reaches max size,
  - logs stored on a separate partition,
  - and running a long-term log retaining system.
- Logs are a treasure-trove of useful information and you should retain them as long as possible as they can help you diagnose an intrusion *post festum.* This is not as good as catching an intrusion on time, but beats not knowing about it or how to remove a root-kit.

Recording more events

- The more auditd records the more power you have to detect problems.
- This means that it is well worth your while to expand auditd's remit to more and more events to log.
- The first thing to enable is to detect any kernel module loading and unloading.
- Compromising the kernel is enoguh to compromise nearly all of the system's security as it is the kernel we trust to enforce our basic security assumptions.
- Loading a malicious module can compromise the kernel in this way.
- Therefore, logging all kernel modification events is very important.

Recording log alterations

- No sensible attacker will permit the system to retain traces of their intrusion.
- Partially as an anti-forensics measure, and partially because if you do not know what the attacker did, you will have a much harder time fixing it.
- Therefore, it is *very* important to log any attempt to modify log files aspecially those that recording login attempts and successes as the attacker is just about guaranteed to appear in those.
- Along these lines, the configuration of autid should be made immutable so as to stop any user from altering it to be blind to their intrustion attempts.

Auditd and time

- System time is an important resource. A lot of security tokens are timed and a lot of protocols rely on system time to derive cryptographic material.
- Therefore, attempts to change system time (which with a modern ntp-based system shouldn't reall ever happen) should be carefully logged as signs of possible malfeasance.
- It should be noted that there is unfortunately a lot of methods to change system time in POSIX systems and our audit will have to keep track of all of them.
- Any that's left out could be a blind spot for the system which is not something we can afford. You will learn which methods exist and how to track them during the labwork.

Auditing security changes

- Auditd should be configured to detect and report any attempt to change file permissions by any process.
- Attempts to alter permissions may indicate dangerous activity, so uses of setxattr, removexattr, chown, fchown, fchownat, chmod, fsetxattr, fchmod, lsetxattr, fremovexattr, lchown, fchmodat, and lremovexattr should be logged.
- This extensive list shows why it is important to either have a guide at hand or to have a complete understanding of system internals.
- Missing any of this somewhat bewildering array of options is potentail blind spot.

Auditing security changes

- All file deletions should be tracked as they may represent anti-forensic measures. If the file itself cannot be preserved, trace of it existing can be.
- Anything that changes any of the network parameters of the system should be carefully logged.
- Any attempts to change logs of process and session initiation should be logged.
- Any attempt to change information about users or groups should be logged: this is a high-priority task as these should be rare enough that there is really no cost in logging them.
- Mandatory access controls that are a part of SELinux controls should only be changed by administrators and any changes or attempts to change them should be logged.

Admin auditing

- CIS CCE-27461-3
- Any action taken by an administrator should be logged
- This is pairs particularly well with the policy with disallowing the direct root shell and instead only elevating privilege for those tasks that require it using the sudo command.
- Not only does this help with securiy, such administartion logging helps with accountability, as admin behavior can be subjected to scrutiny after the fact
- This can help determine fault in case of mistakes or detect malicious behavior.
- Similarly, any privileged command with a setuid or setgid bit set should be placed under scrutiny, both for accountability, again, and because such commands are prime candidates for escalation-of-privilege attacks.