© SANS Institute 2021

AUD507 | AUDITING & MONITORING NETWORKS, PERIMETERS, AND SYSTEMS

507.3 Advanced UNIX Auditing and Monitoring

2021

SANS

THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

August 10, 2021

© SANS Institute 2021

© 2021 Risenhoover Consulting, Inc. All rights reserved to Risenhoover Consulting, Inc. and/or SANS Institute.

PLEASE READ THE TERMS AND CONDITIONS OF THIS COURSEWARE LICENSE AGREEMENT ("CLA") CAREFULLY BEFORE USING ANY OF THE COURSEWARE ASSOCIATED WITH THE SANS COURSE. THIS IS A LEGAL AND ENFORCEABLE CONTRACT BETWEEN YOU (THE “USER”) AND SANS INSTITUTE FOR THE COURSEWARE. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU.

With this CLA, SANS Institute hereby grants User a personal, non-exclusive license to use the Courseware subject to the terms of this agreement. Courseware includes all printed materials, including course books and lab workbooks, as well as any digital or other media, virtual machines, and/or data sets distributed by SANS Institute to User for use in the SANS class associated with the Courseware. User agrees that the CLA is the complete and exclusive statement of agreement between SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA.

BY ACCEPTING THIS COURSEWARE,USER AGREES TO BE BOUND BY THE TERMS OF THIS CLA. BY ACCEPTING THIS SOFTWARE, USER AGREES THAT ANY BREACH OF THE TERMS OF THIS CLA MAY CAUSE IRREPARABLE HARM AND SIGNIFICANT INJURY TO SANS INSTITUTE, AND THAT SANS INSTITUTE MAY ENFORCE THESE PROVISIONS BY INJUNCTION (WITHOUT THE NECESSITY OF POSTING BOND) SPECIFIC PERFORMANCE, OR OTHER EQUITABLE RELIEF.

If User does not agree, User may return the Courseware to SANS Institute for a full refund, if applicable.

User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of the Courseware, in any medium whether printed, electronic or otherwise, for any purpose, without the express prior written consent of SANS Institute. Additionally, User may not sell, rent, lease, trade, or otherwise transfer the Courseware in any way, shape, or form without the express written consent of SANS Institute.

If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this Courseware.

SANS acknowledges that any and all software and/or tools, graphics, images, tables, charts or graphs presented in this Courseware are the sole property of their respective trademark/registered/copyright owners, including:

AirDrop, AirPort, AirPort Time Capsule, Apple, Apple Remote Desktop, Apple TV, App Nap, Back to My Mac, Boot Camp, Cocoa, FaceTime, FileVault, Finder, FireWire, FireWire logo, iCal, iChat, iLife, iMac, iMessage, iPad, iPad Air, iPad Mini, iPhone, iPhoto, iPod, iPod classic, iPod shuffle, iPod nano, iPod touch, iTunes, iTunes logo, iWork, Keychain, Keynote, Mac, Mac Logo, MacBook, MacBook Air, MacBook Pro, Macintosh, Mac OS, Mac Pro, Numbers, OS X, Pages, Passbook, Retina, Safari, Siri, Spaces, Spotlight, There’s an app for that, Time Capsule, Time Machine, Touch ID, Xcode, Xserve, App Store, and iCloud are registered trademarks of Apple Inc.

PMP® and PMBOK® are registered trademarks of PMI.

SOF-ELK® is a registered trademark of Lewes Technology Consulting, LLC. Used with permission.

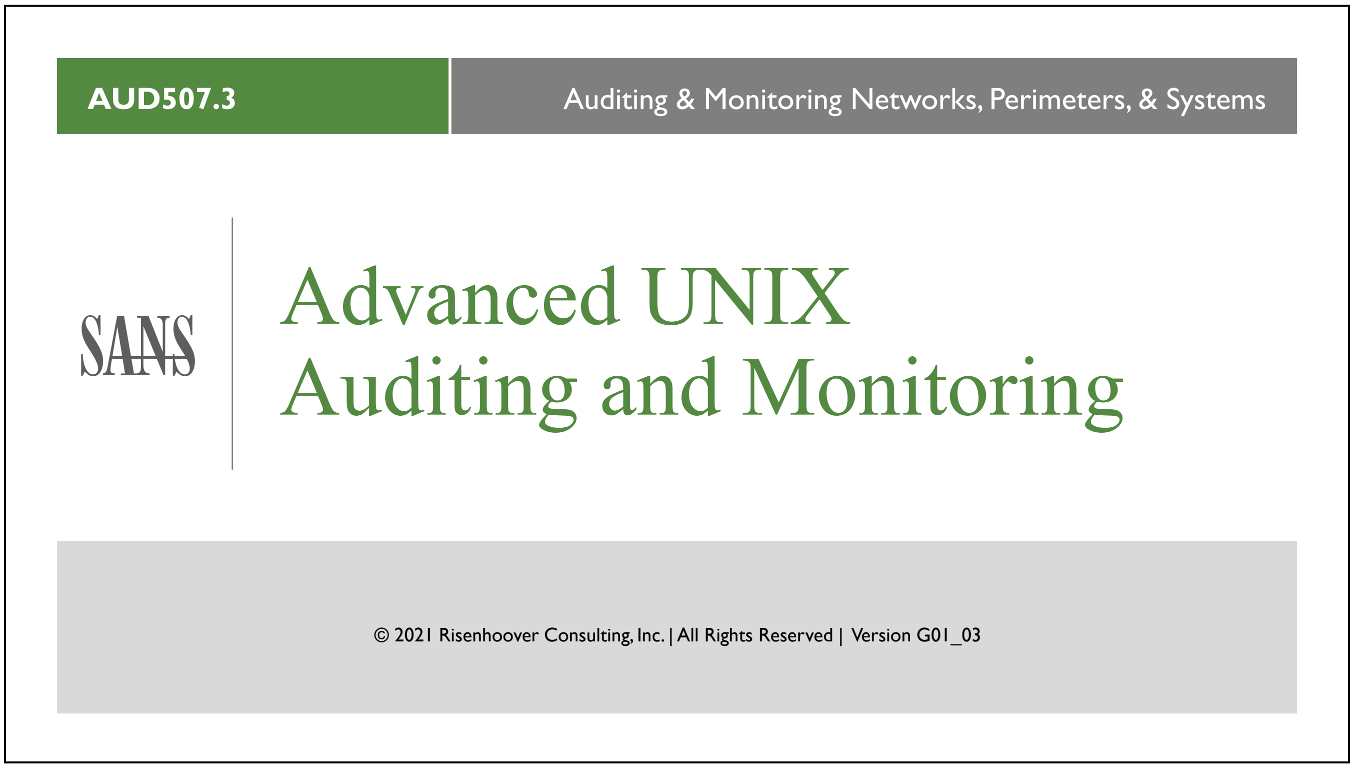
SIFT® is a registered trademark of Harbingers, LLC. Used with permission.

Governing Law: This Agreement shall be governed by the laws of the State of Maryland, USA.

August 10, 2021

Technet24

© SANS Institute 2021



AUD507.3

Auditing & Monitoring Networks, Perimeters, & Systems

SANS

Advanced UNIX Auditing and Monitoring

© 2021 Risenhoover Consulting, Inc. | All Rights Reserved | Version G01_03

Welcome to section 3 of the SANS AUD507 course! This course is written, maintained, and frequently taught by Clay Risenhoover. I am always looking for ways to improve this courseware. If you have questions or suggestions for how to improve the course, or if you need any additional materials referenced during the class, please let me know. If you find errors or inaccuracies in the course books, I encourage you to pass those on to me. You can email me at clay@risenhooverconsulting.com. Please put either "SANS" or "AUD507" in the subject line, to ensure I see the email.

The entire content of this and every other volume in this course is © 2021 Risenhoover Consulting, Inc.

August 10, 2021

August 10, 2021

© SANS Institute 2021

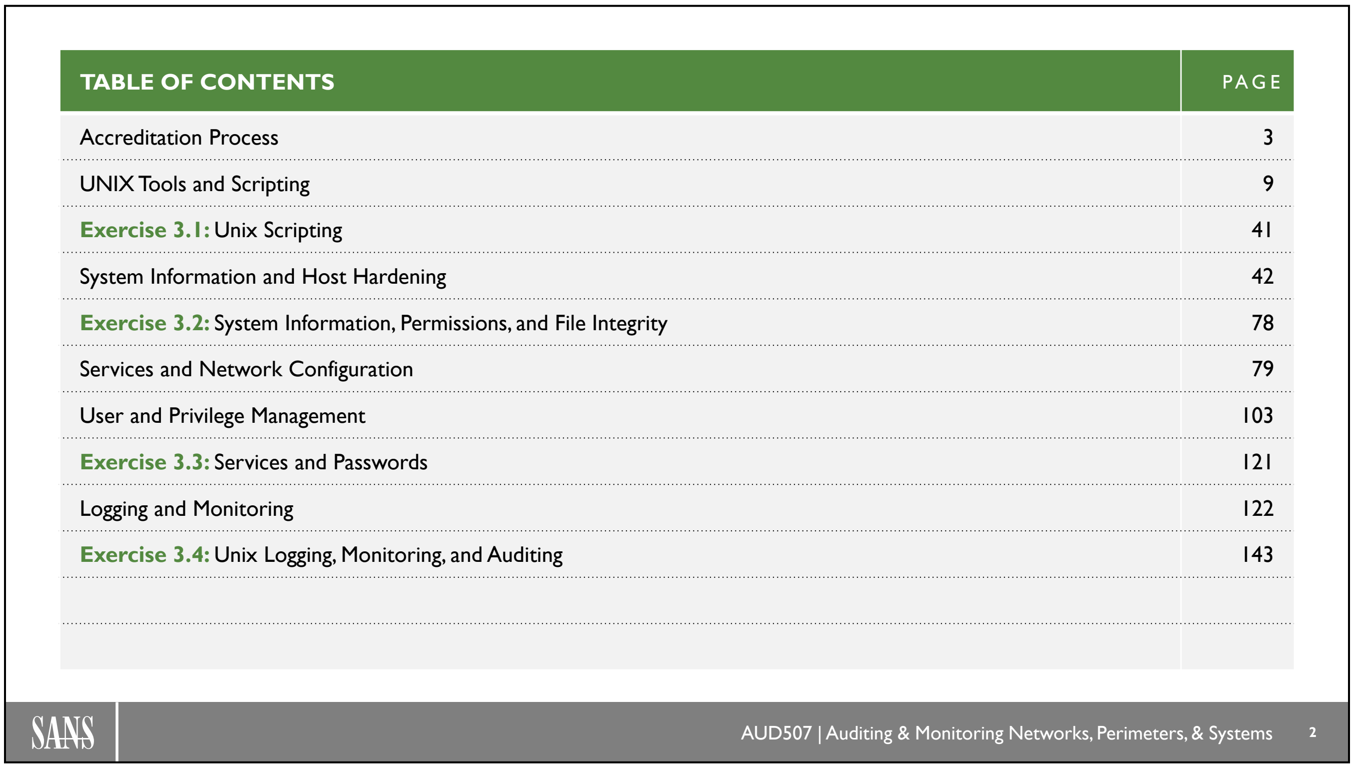


TABLE OF CONTENTS

This page intentionally left blank.

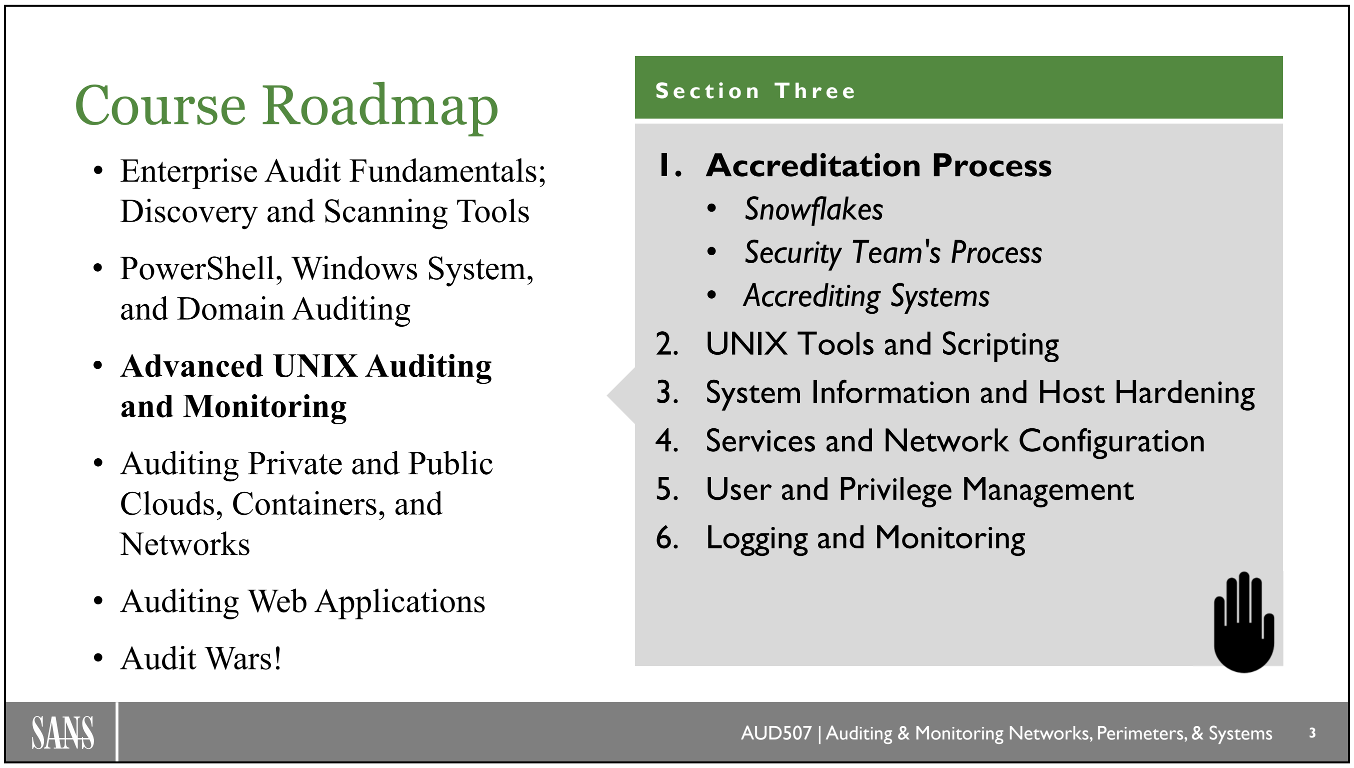
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Course Roadmap

- Enterprise Audit Fundamentals; Discovery and Scanning Tools
- PowerShell, Windows System, and Domain Auditing
- **Advanced UNIX Auditing and Monitoring**
- Auditing Private and Public Clouds, Containers, and Networks
- Auditing Web Applications
- Audit Wars!

Section Three

1. **Accreditation Process**
   - *Snowflakes*
   - *Security Team's Process*
   - *Accrediting Systems*
2. UNIX Tools and Scripting
3. System Information and Host Hardening
4. Services and Network Configuration
5. User and Privilege Management
6. Logging and Monitoring

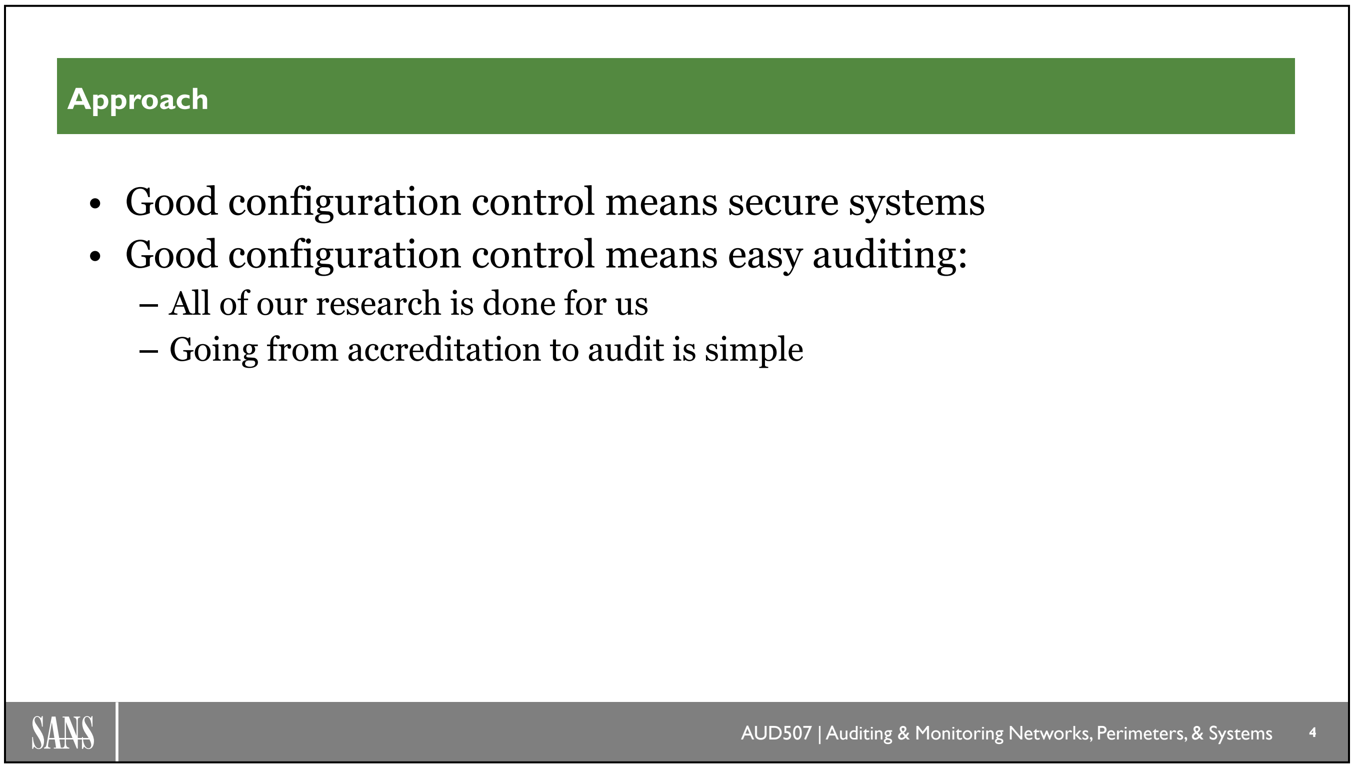
SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 3

In this section, we will cover everything from basic UNIX concepts and configuration, clear through to advanced auditing techniques, stopping off at security configuration, maintenance, logging, and a how-to for a fairly comprehensive baseline audit program. We do expect that you have a basic working knowledge of UNIX coming into this course, just as we did with Windows.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Approach

- Good configuration control means secure systems
- Good configuration control means easy auditing:
  - All of our research is done for us
  - Going from accreditation to audit is simple

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 4

The idea behind an accreditation process is that there is some strong form of configuration control implemented within the organization. This configuration control could be to the point where every system is built using a default image, and then whichever individual options need to be turned on for the system to perform its function are configured. When it comes to UNIX systems, this is actually far more work than it is under operating systems like Windows.

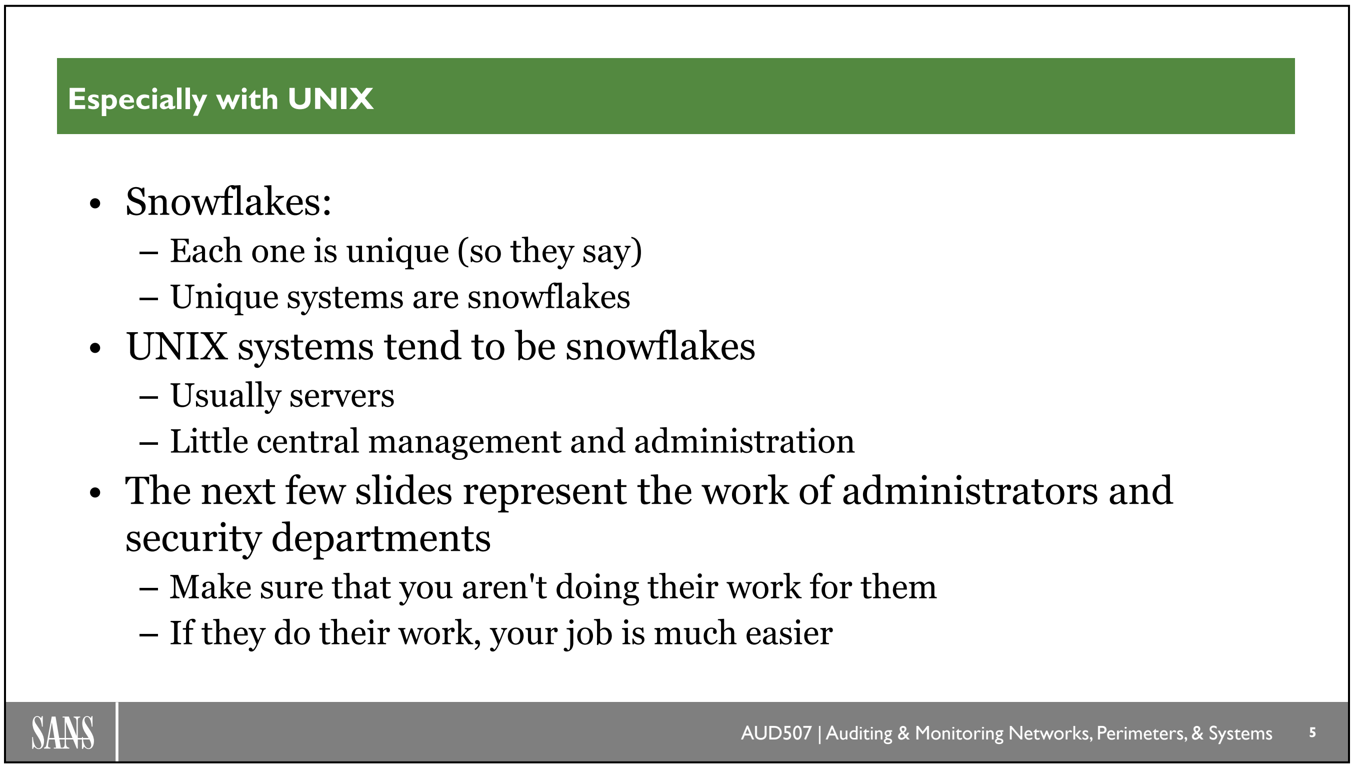
With that in mind, it might be better to apply a security accreditation process whereby we require that a somewhat formal risk assessment of the system be performed according to a set of guidelines that have been laid out by the organization. The system administrator, in the process of completing the accreditation process, fills out an accreditation form that includes this risk assessment. The resulting form becomes the basis for our audit of that system because, if the system is permitted to be brought into production after the accreditation process, then the accreditation forms must contain the relevant information about that system from an information flow perspective!

August 10, 2021

Everything that is discussed in this section can be applied in a Windows environment, but we tend to have much stronger configuration controls in the Windows world. Workstations, for example, should be built from a standard image and software deployed via group policy and the domain. In UNIX, however, the nature of the systems is quite different. For this reason, we will tackle accreditation in the context of UNIX systems.

© 2021 Risenhoover Consulting, Inc.
August 10, 2021
Technet24

© SANS Institute 2021



Especially with UNIX

- Snowflakes:
  - Each one is unique (so they say)
  - Unique systems are snowflakes
- UNIX systems tend to be snowflakes
  - Usually servers
  - Little central management and administration
- The next few slides represent the work of administrators and security departments
  - Make sure that you aren't doing their work for them
  - If they do their work, your job is much easier

SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 5

This information regarding accreditation is of equal value for all kinds of systems, but we chose to talk about it in the UNIX material rather than the Windows material because our UNIX systems tend to be less standardized than our Windows desktops. The term that's used to describe systems that vary from the norm within an environment is "snowflakes." Likely, you can derive the meaning, knowing that it is said that every snowflake is unique. Although a few snowflakes are quite beautiful, you might also know, depending on where in the world you come from, that when a bunch of snowflakes gang up on you, you're in for a bad day.

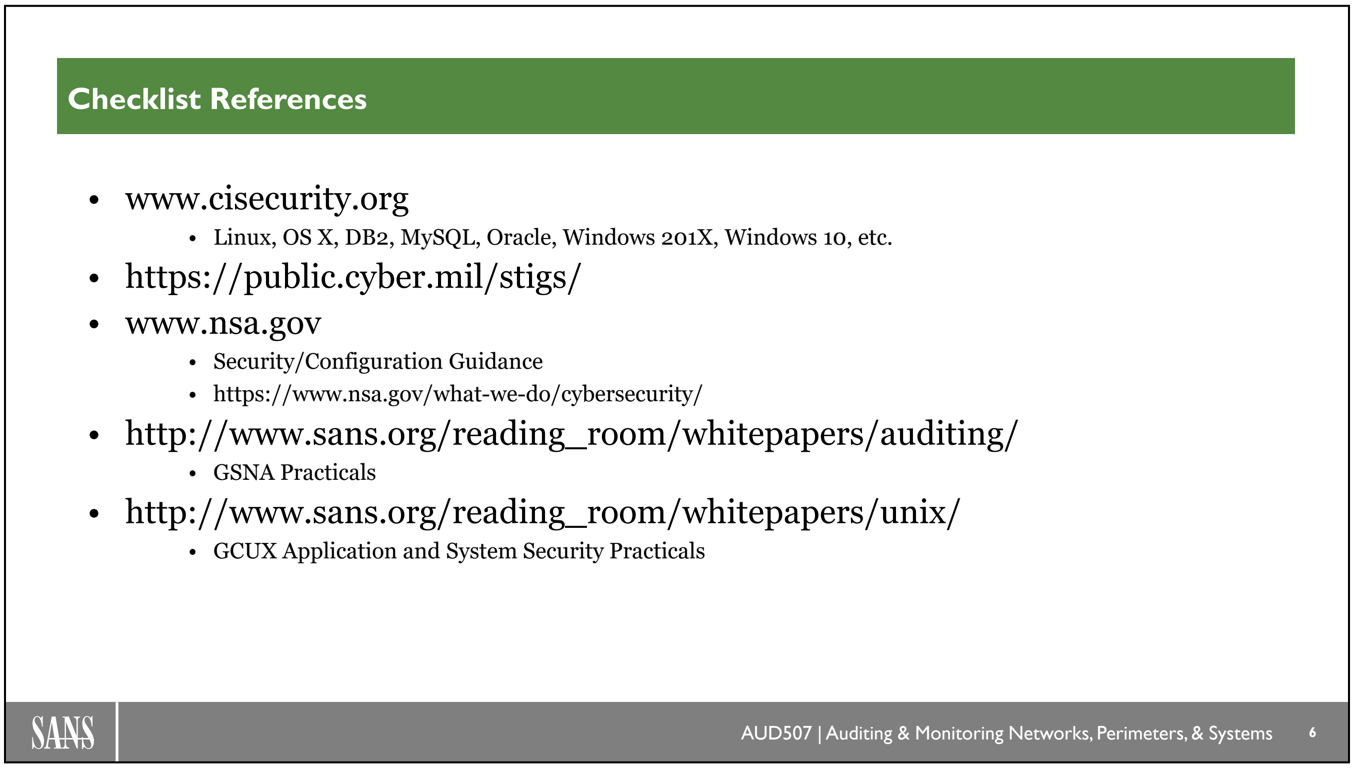
The same is true in our networks. One or two snowflakes, no problem. What happens when all of the systems are snowflakes, though? That's a real nightmare! UNIX systems tend to be of this variety because these systems are usually deployed as servers, and each server has a different role. This means that they will inherently be different. UNIX system administrators also tend to feel a greater sense of control over their systems, perhaps even a territorial inclination, which tends to mean that these systems will become more and more customized over time.

This isn't necessarily a bad thing, but it does mean that we need to think seriously about how we can manage these systems, especially from the point of view of auditing. The concept of using an accreditation system is exactly what we need. Keep in mind that, as we start, we are describing the process that the system, network, and security engineers should be following. Auditors step into the process several slides from now. We would suggest that you recommend that the process we describe is followed; just make sure you don't end up doing someone else's job!

August 10, 2021

August 10, 2021

© SANS Institute 2021



Checklist References

- www.cisecurity.org
  - Linux, OS X, DB2, MySQL, Oracle, Windows 201X, Windows 10, etc.
- https://public.cyber.mil/stigs/
- www.nsa.gov
  - Security/Configuration Guidance
  - https://www.nsa.gov/what-we-do/cybersecurity/
- http://www.sans.org/reading_room/whitepapers/auditing/
  - GSNA Practicals
- http://www.sans.org/reading_room/whitepapers/unix/
  - GCUX Application and System Security Practicals

SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 6

If your organization already has a well-defined standard for the deployment of secure UNIX systems, that would be the starting point for the development of an accreditation worksheet. If you do not currently have well-defined standards, someone, likely the security officer, should use an example such as those on the slide to derive a list of items that are requirements for security within your organization. You'd prefer that this list be abstracted from a specific operating system brand so that it can be applied to any operating system or at least a class of operating systems.

The Center for Internet Security has been releasing benchmarks and tools for many years that allow you to audit systems against their consensus benchmarks. These benchmarks are very well documented and represent the minimum that should be done to lock down a system.

The Defense Information Systems Agency (DISA) has a repository of security review requirements documents and checklists for all US DoD and US government sites that is pretty useful and updated fairly frequently. This is definitely a site worth looking at for an accreditation source.

You will also want to check out many of the practical assignments written as practicals for the UNIX Track Certification. There are checklists there for most flavors of UNIX and many UNIX applications.

August 10, 2021

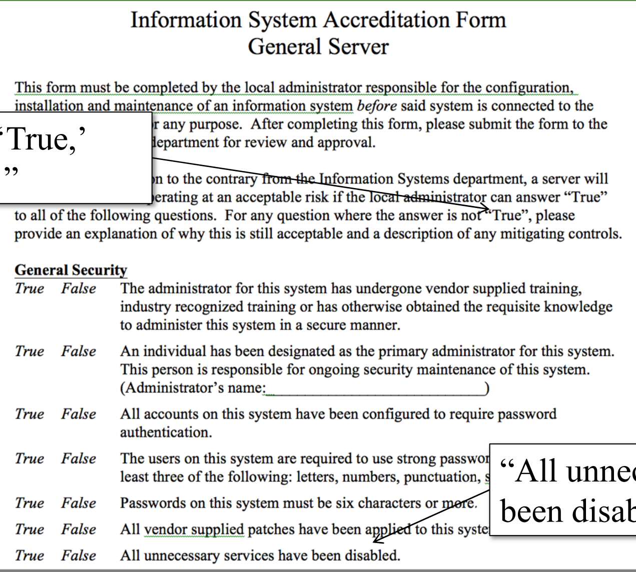
© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021

Risk Assessment Questionnaire



Information System Accreditation Form
General Server

This form must be completed by the local administrator responsible for the configuration, installation and maintenance of an information system *before* said system is connected to the r any purpose. After completing this form, please submit the form to the department for review and approval.

n to the contrary from the Information Systems department, a server will erating at an acceptable risk if the local administrator can answer "True" to all of the following questions. For any question where the answer is not "True", please provide an explanation of why this is still acceptable and a description of any mitigating controls.

General Security

True	*False*	The administrator for this system has undergone vendor supplied training, industry recognized training or has otherwise obtained the requisite knowledge to administer this system in a secure manner.
True	*False*	An individual has been designated as the primary administrator for this system. This person is responsible for ongoing security maintenance of this system. (Administrator's name:_______________________)
True	*False*	All accounts on this system have been configured to require password authentication.
True	*False*	The users on this system are required to use strong passwor least three of the following: letters, numbers, punctuation, s
True	*False*	Passwords on this system must be six characters or more.
True	*False*	All vendor supplied patches have been applied to this syste
True	*False*	All unnecessary services have been disabled.

"If you can't answer 'True,' you must explain it…"

"All unnecessary services have been disabled."

An example of what this type of accreditation or risk assessment worksheet might look like is shown here.

Consider the short selection of questions that we've included on the slide. Notice the last two, for example. True or false: All vendor-supplied patches have been applied to the system. True or false: All unnecessary services have been disabled. The worksheet is generic enough to be applied to any system you might have in your enterprise, yet simultaneously provides the administrator with what amounts to instructions on the proper configuration, which will allow him to get the system connected.

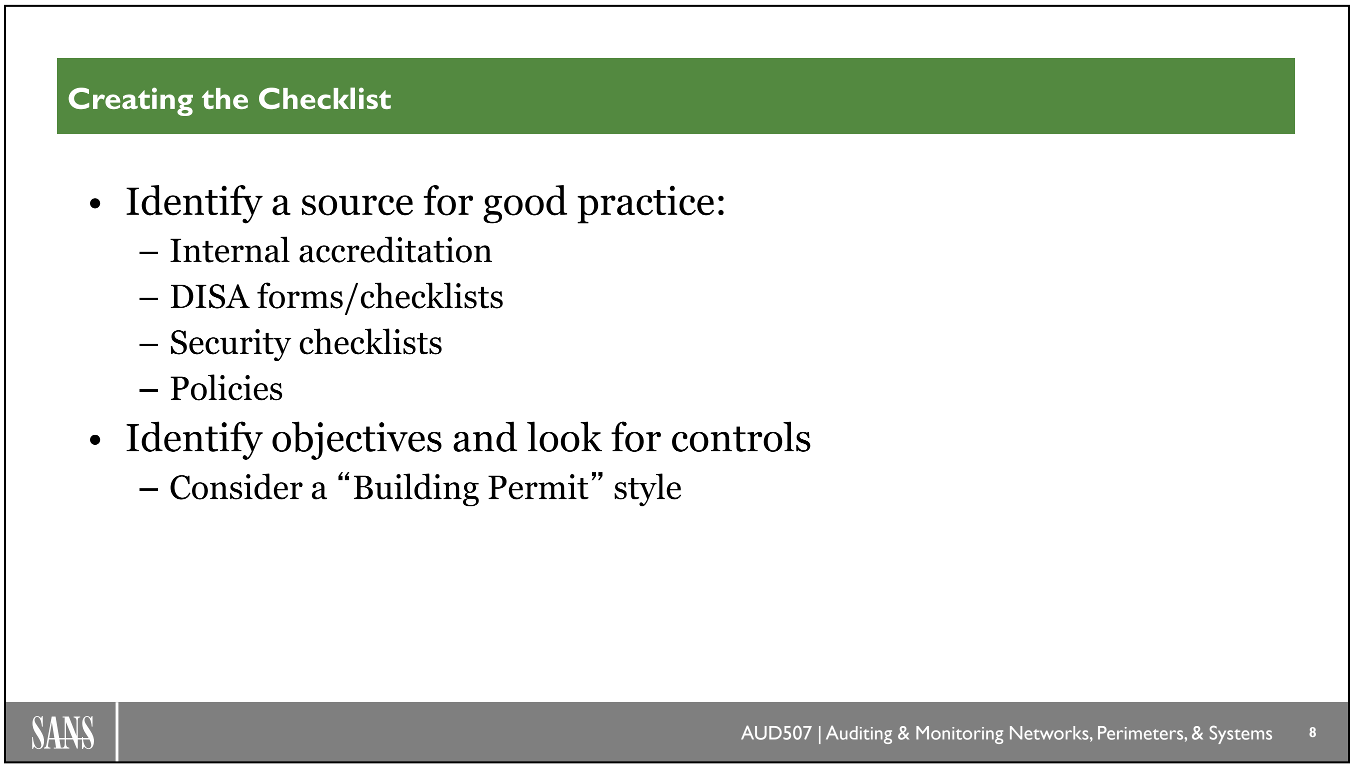
If the administrator answers any question "False," he must then explain how that issue has been otherwise remediated. The completed form is then submitted to the security department for review by someone authorized by the security officer or by the security officer himself. Unless there has been some previous experience with this administrator that would require a spot check and provided there are no unusual issues in the questionnaire itself, the person evaluating the risk assessment worksheet can issue a provisional approval. Of course, these individuals may perform spot checks whenever they deem it necessary, much as a building inspector would tell you to call him for an inspection before you put sheetrock up, allowing him to see the electrical work.

Finally, when it is time for us to perform an audit, ultimately accrediting the system, we simply pull the customized risk assessment worksheet that *has already been approved!* In other words, rather than having to create a customized checklist for this system, we can use the approved sheet that defines acceptable risk for that system. Again, like a building inspector, I am no longer concerned with what the letter of the law is. I now simply verify that the administrator has followed the approved "building plans" with approved "variances" from the standard requirements.

Of course, you can always provide observations and other feedback to the administrators and management, allowing the entire process to become refined over time.

August 10, 2021

© SANS Institute 2021



Creating the Checklist

- Identify a source for good practice:
  - Internal accreditation
  - DISA forms/checklists
  - Security checklists
  - Policies
- Identify objectives and look for controls
  - Consider a "Building Permit" style

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 8

So, then, creating a checklist to audit a system turns out to be a pretty simple task once you have a good idea of how that system operates. This is again why the "Research" phase of an audit is probably the most critical! For our sources, we have identified several possible organizations that could assist with free information, in addition to our own organizational documentation and policies.

After we decide which sources we will use for good practice for an audit, we now need to identify various objectives and then find controls that are in place to meet the objectives. Simultaneously, we are identifying audit controls that allow us to monitor or check on how the security controls are functioning.

One thing that I've learned works really well is to create something analogous to a "Building Permit" and "Zoning" system. Here's what I mean. To add a detached garage to your property, you would need to go through the local building department. If your property isn't zoned for the type of building that you want to do, you need to apply for a variance. After that's done, you have to submit building plans that demonstrate that you will follow all of the applicable building codes. Does an inspector need to come and check every nail and screw? No. The inspector comes only at major milestones and spot-checks your progress. He uses your approved plans to determine whether or not you are still "legal."

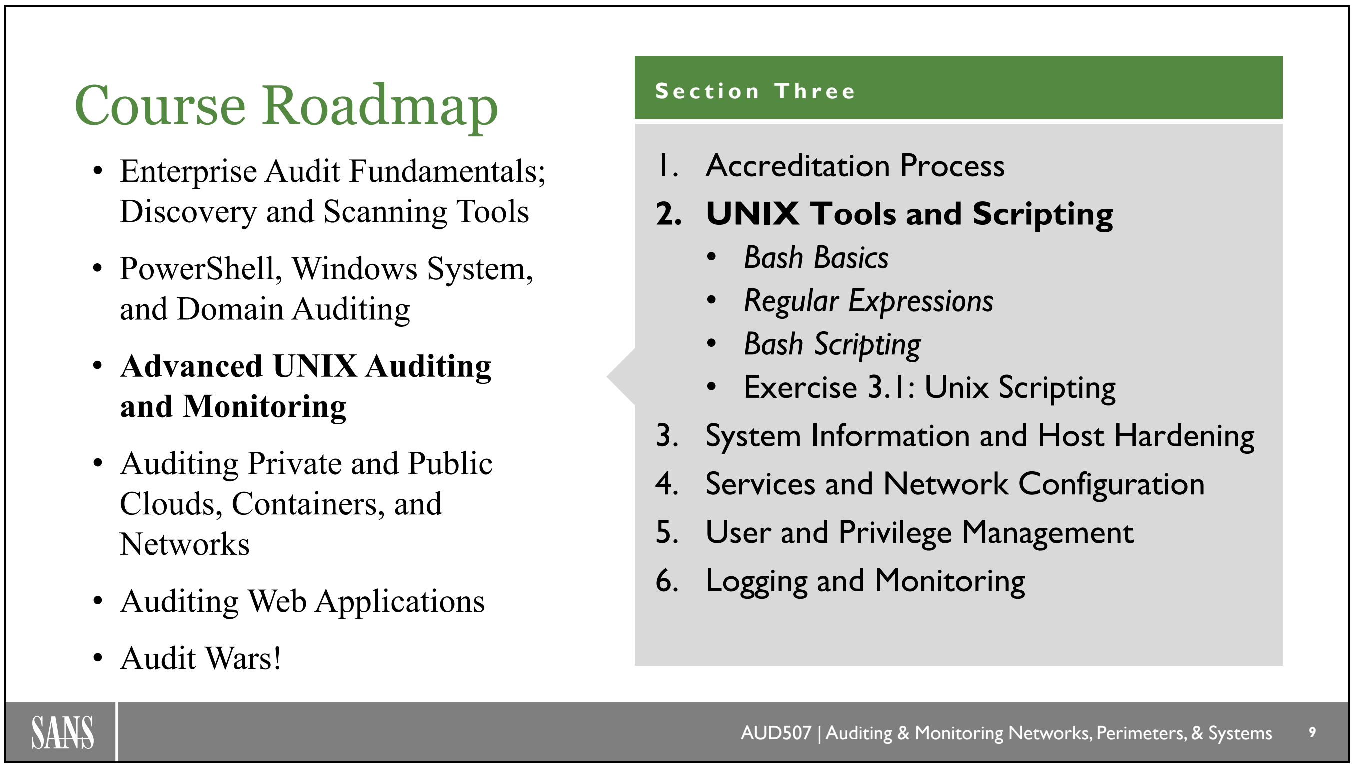
This, it turns out, is a fantastic way to manage compliance and to make auditing of specialized systems, which UNIX systems tend to be, very easy! The administrator must complete a Risk Self-Assessment form and submit it to the security team. For anything that he answers in a way that indicates risk, he must also explain what compensating controls have been put in place. If everything seems reasonable, the security team approves his "blueprints." Now, finally, the auditor's role becomes apparent. When the auditor needs to check compliance, he can get a copy of the checklist that the administrator filled out and that was approved. The system when checked *must* match the sheet that was submitted. If it does, it is operating at an acceptable level of risk.

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Course Roadmap

- Enterprise Audit Fundamentals; Discovery and Scanning Tools
- PowerShell, Windows System, and Domain Auditing
- **Advanced UNIX Auditing and Monitoring**
- Auditing Private and Public Clouds, Containers, and Networks
- Auditing Web Applications
- Audit Wars!

Section Three

1. Accreditation Process
2. **UNIX Tools and Scripting**
   - *Bash Basics*
   - *Regular Expressions*
   - *Bash Scripting*
   - Exercise 3.1: Unix Scripting
3. System Information and Host Hardening
4. Services and Network Configuration
5. User and Privilege Management
6. Logging and Monitoring

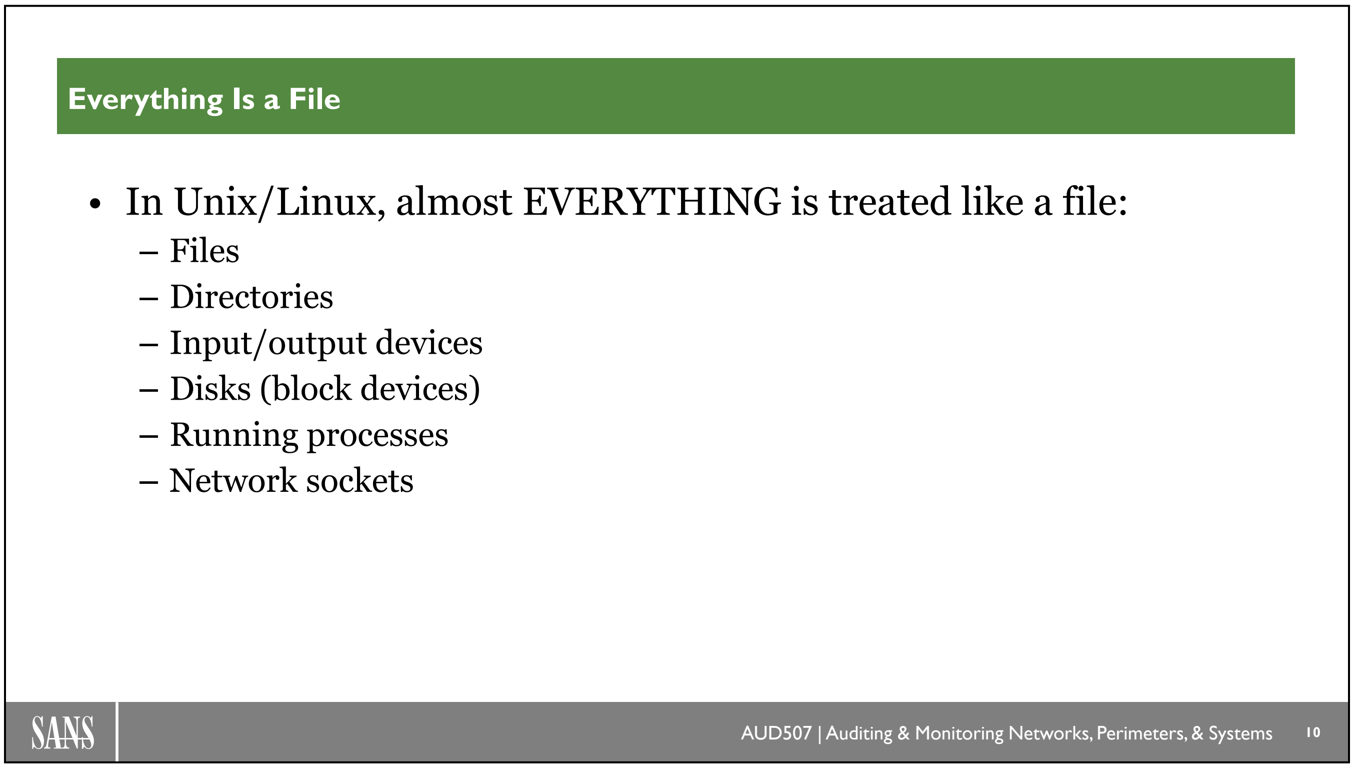
SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 9

Our next module on our path to auditing UNIX systems will bring us into the wonderful world of scripting. In this section, we cover the basics of UNIX scripting and use this as a foundation for the remainder of the section, building a fairly good set of audit scripts over the course of the material.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Everything Is a File

- In Unix/Linux, almost EVERYTHING is treated like a file:
  - Files
  - Directories
  - Input/output devices
  - Disks (block devices)
  - Running processes
  - Network sockets

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 10

Unix-like operating systems treat most objects which can be managed by the operating system as files. This homogenous approach to resource management allows us to use the same tools to interact with many different types of operating system objects.

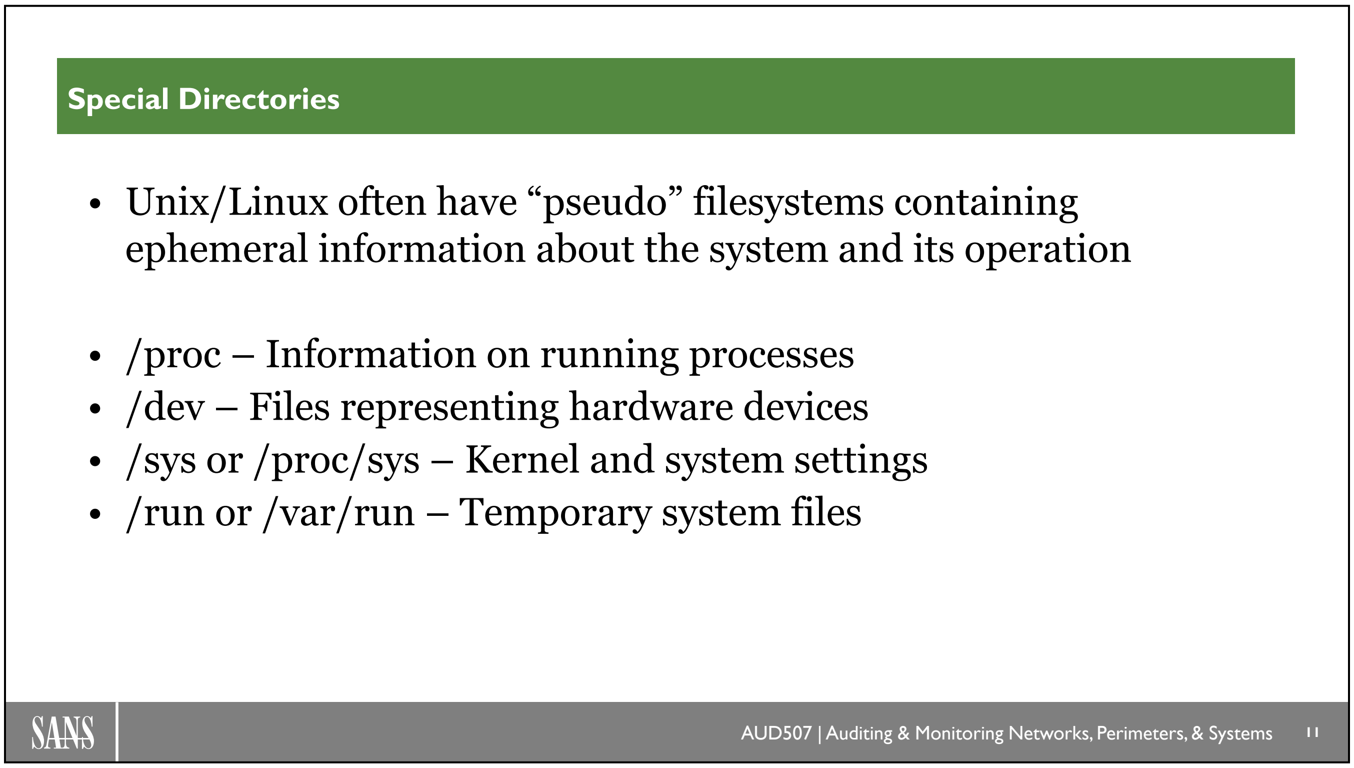
As we work through the material, we will explore tools which can report on these different "file" types in some really useful ways.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Special Directories

- Unix/Linux often have "pseudo" filesystems containing ephemeral information about the system and its operation

- /proc – Information on running processes
- /dev – Files representing hardware devices
- /sys or /proc/sys – Kernel and system settings
- /run or /var/run – Temporary system files

SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 11

Linux and Unix systems will often have special directories used to present administrators and users with information about the running system.

The /proc filesystem is used to maintain data about running processes. Most systems maintain a directory under proc for each running process. These directories contain virtual files with information about the command line used to run the program, any files it has open, its memory state, and much more.

The /dev/filesystem contains virtual files which represent the hardware installed on a device. For instance, on a Linux laptop, /dev/sda might represent the first SATA hard drive installed in the system. The device filenames are used when running commands to interact with a specific piece of hardware.

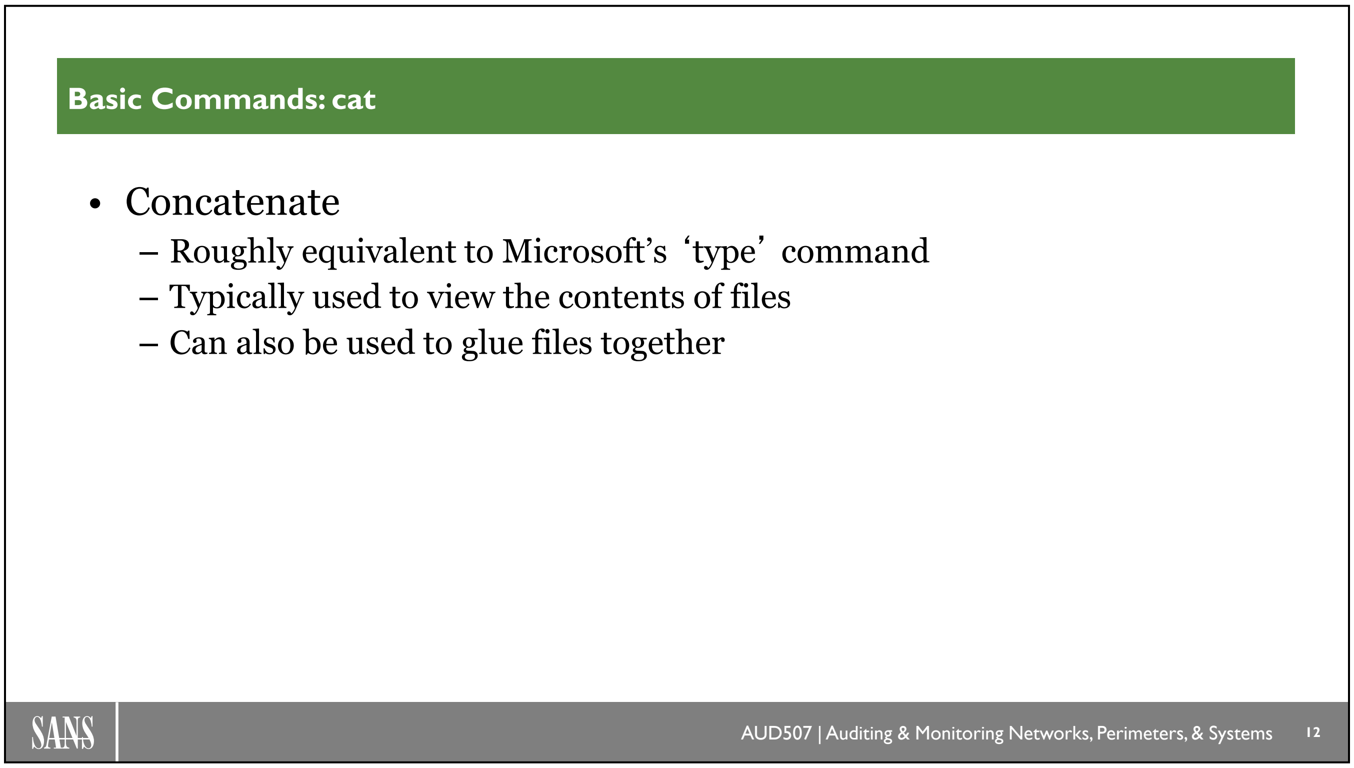
The /sys (or /proc/sys) filesystem is used to expose Linux/Unix kernel and system settings. Administrators can sometimes change the settings on the system by changing the content of the virtual files in this filesystem. On my Debian Linux web server, the setting for whether the host will act as an IPv4 router is at /proc/sys/net/ipv4/ip_forward. On some systems, these settings are controlled using the "sysctl" command.

The /run (or /var/run) directory is used to store temporary system files which do not need to survive a reboot. A common example is that many Unix/Linux services will write a file to this directory which contains the running executable's process ID (PID) number to assist in managing the executable.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Basic Commands: cat

- Concatenate
  - Roughly equivalent to Microsoft's 'type' command
  - Typically used to view the contents of files
  - Can also be used to glue files together

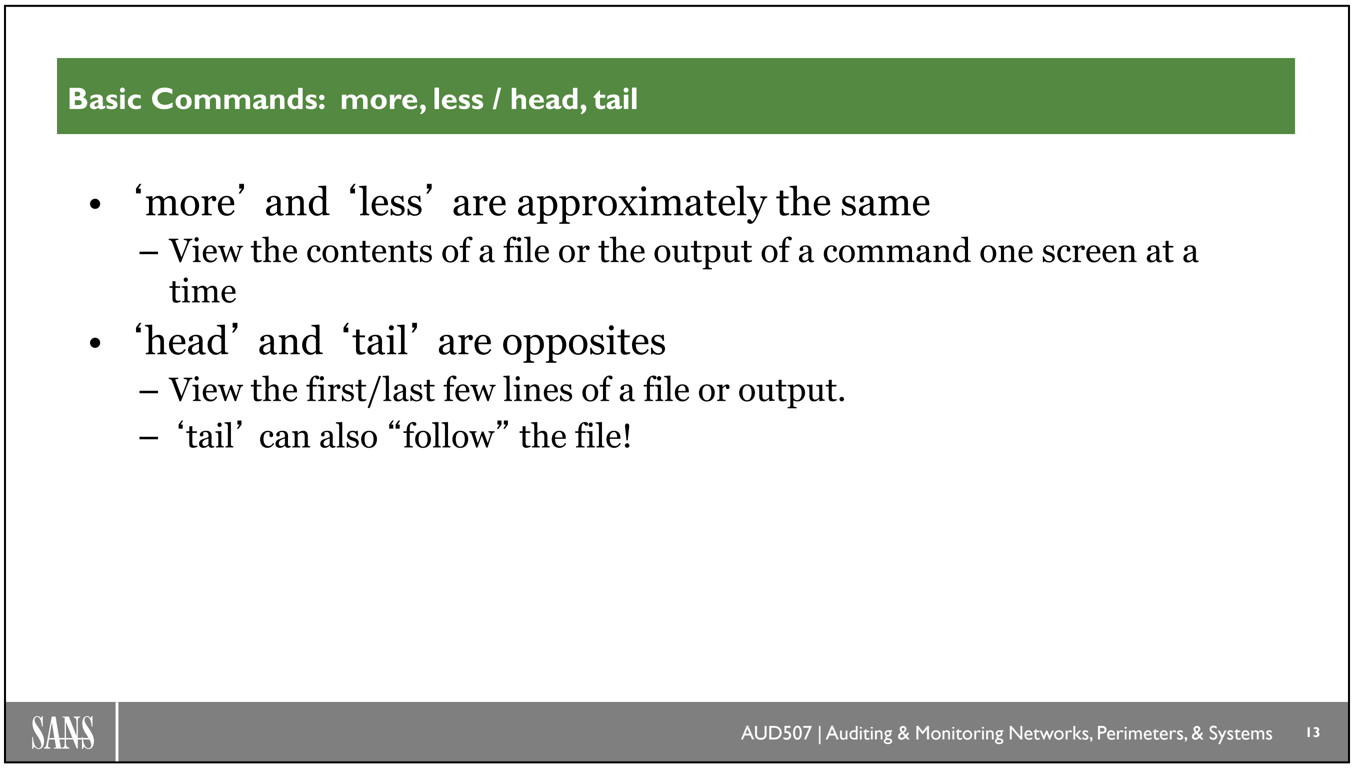
The 'cat' command is an abbreviation for "concatenate" and is roughly equivalent to the Microsoft 'type' command. The 'cat' utility can be used to quickly view the contents of a file, send the contents of a file through some other utility as input, or glue files together, which is the actual definition of concatenate. This tool is useful on its own as long as the contents of the file are less than one screen in length or if what we want to see is at the end of the file. If the file is longer, there are a few other tools we should know about.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Basic Commands: more, less / head, tail

- 'more' and 'less' are approximately the same
  - View the contents of a file or the output of a command one screen at a time
- 'head' and 'tail' are opposites
  - View the first/last few lines of a file or output.
  - 'tail' can also "follow" the file!

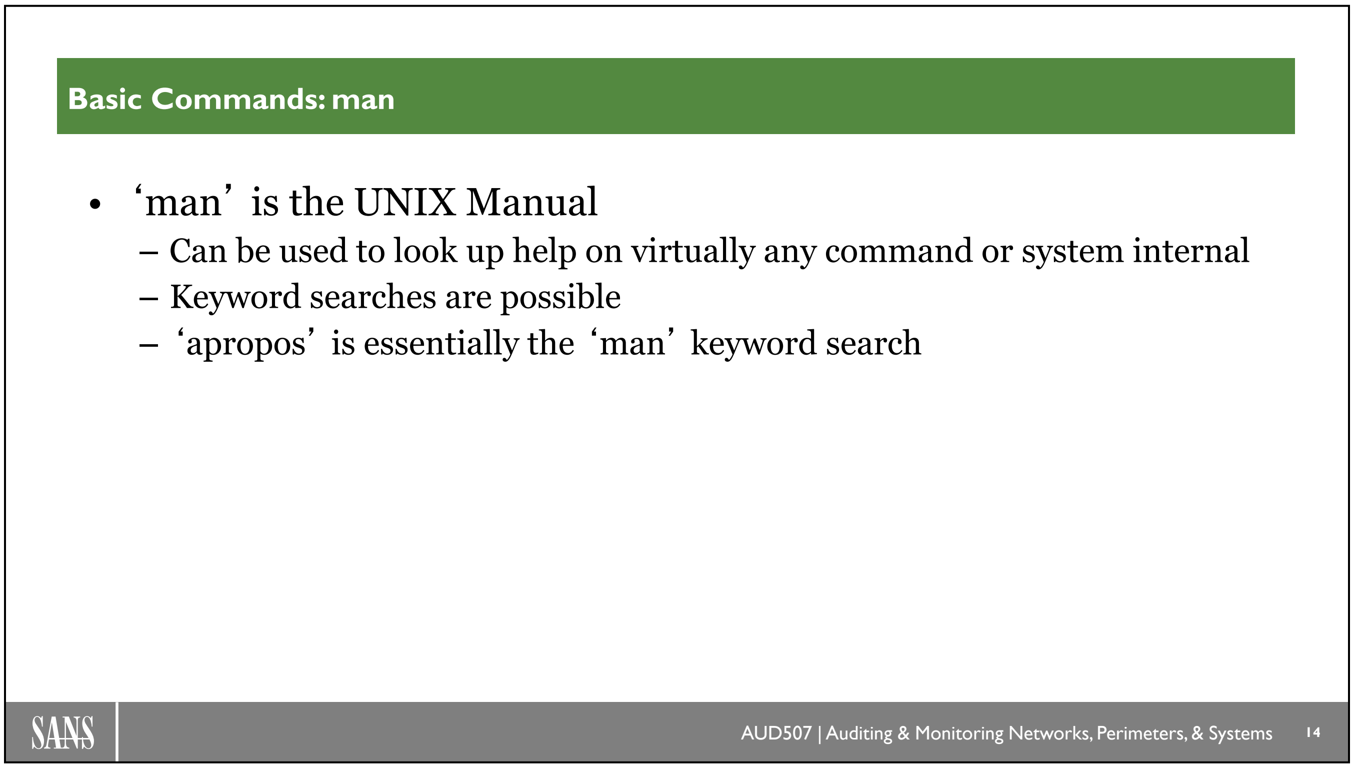
Not all files will be small enough to view using 'cat'. To solve this problem, we have a variety of nifty tools to handle these situations.

The 'more' and 'less' tools are more or less the same. They are both used to view text files or command output one page at a time. If you are using these tools and would like to advance to the next page, pressing the spacebar will skip you ahead. You can also press the Enter or Return key to move ahead one line. In many cases, you might also want to use the 'b' key to go back one page. When you are finished viewing the file or output and want to leave 'more' or 'less', just press the 'q' key to quit.

Our other two tools, 'head' and 'tail', are opposites of one another. These utilities can be used to see the head of a file, typically the first five to ten lines, or the tail of a file, the last five to ten lines. You can, of course, specify exactly how many lines you would like to see; you can even specify the line offset that you would like either tool to start displaying from, relative to the head or tail of the file. 'tail' has one additional feature that is incredibly useful when it comes to monitoring logs. It is possible to tell 'tail' to "follow" a file, which means that it will display the last few lines of the file in question and then begin to wait for more text to be written into the file. As new lines are appended to the file, 'tail' will display them for us. The '-f' option is used to put 'tail' into "follow" mode.

August 10, 2021

© SANS Institute 2021



Basic Commands: man

- 'man' is the UNIX Manual
  - Can be used to look up help on virtually any command or system internal
  - Keyword searches are possible
  - 'apropos' is essentially the 'man' keyword search

If you are new to UNIX or if it has been a long time since you actually used UNIX, your head might be spinning already! It turns out that you really don't have to remember all of this stuff. Very early on, the creators and developers of UNIX decided that it would become extremely cumbersome to create a printed manual. In fact, you can buy UNIX manuals for commercial Unices, but both the free and commercial distributions have maintained electronic manuals as well. These manuals are accessible from the command line in UNIX by using the 'man' command. There are also a number of GUI-based manual page viewers like 'Xman.'

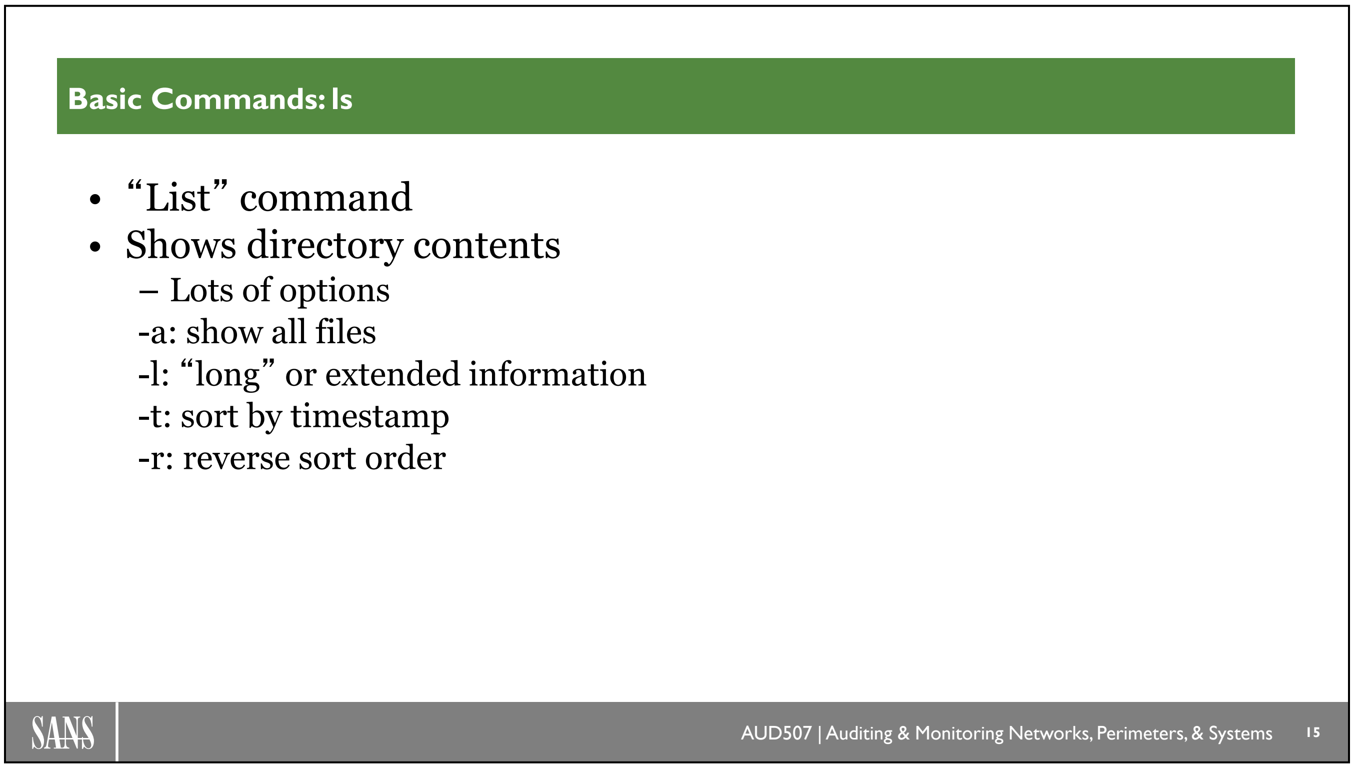
The 'man' command might not be useful if you had to know the name of the command that you wanted help on first. To this end, you can also perform keyword searches using either the 'man -k' command or the 'apropos' command. Either of these will allow you to enter keywords to search for, and the manual system will report any potentially useful manual pages installed on the system.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Basic Commands: ls

- "List" command
- Shows directory contents
  - Lots of options

    -a: show all files

    -l: "long" or extended information

    -t: sort by timestamp

    -r: reverse sort order

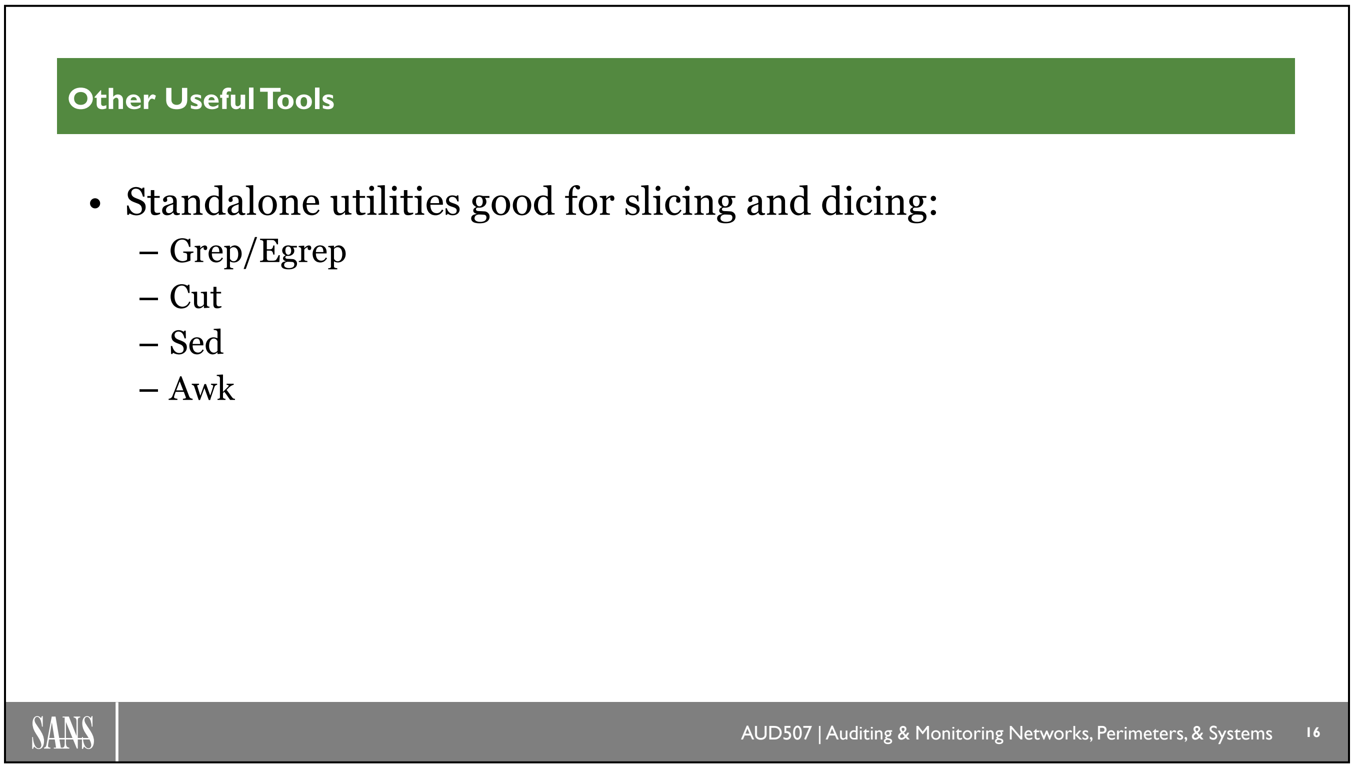
The 'ls' command is probably the command you will run more frequently than any other command at a UNIX command line. Quite simply, 'ls' stands for "list." 'ls' allows you to list all the files that are in the current directory or potentially in the directory that you specify. Like most UNIX tools, 'ls' has lots and lots of possible options. Even if you've been using UNIX for some time, please take some time during the exercise section to actually leaf through the manual page for 'ls'; chances are that even seasoned administrators will find some handy feature that they were not previously aware of or completely forgot about.

Some very useful and common options, especially for auditors or incident handlers, are the '-a' option to show all files (including hidden files—a file is considered "hidden" if the name begins with a '.'), the '-l' option (to obtain a "long" listing that includes file modification time, permissions, owner, group, link count, etc.), the '-t' option (which lists files according to the modification date), and the '-r' option (which is used to reverse the sort order of the files). The normal sort order for a directory listing is alphabetical.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Other Useful Tools

- Standalone utilities good for slicing and dicing:
  - Grep/Egrep
  - Cut
  - Sed
  - Awk

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 16

Here are a few other handy tools for slicing and dicing the output that comes out of the various other tools we will use to collect information from our systems. All of these tools could easily take up an hour or more of our time, so we're going to give you a very brief introduction to them over the next few slides and suggest some handy ways that they might be used. As we go through the material, we'll use these tools to build up a useful information-collection script.

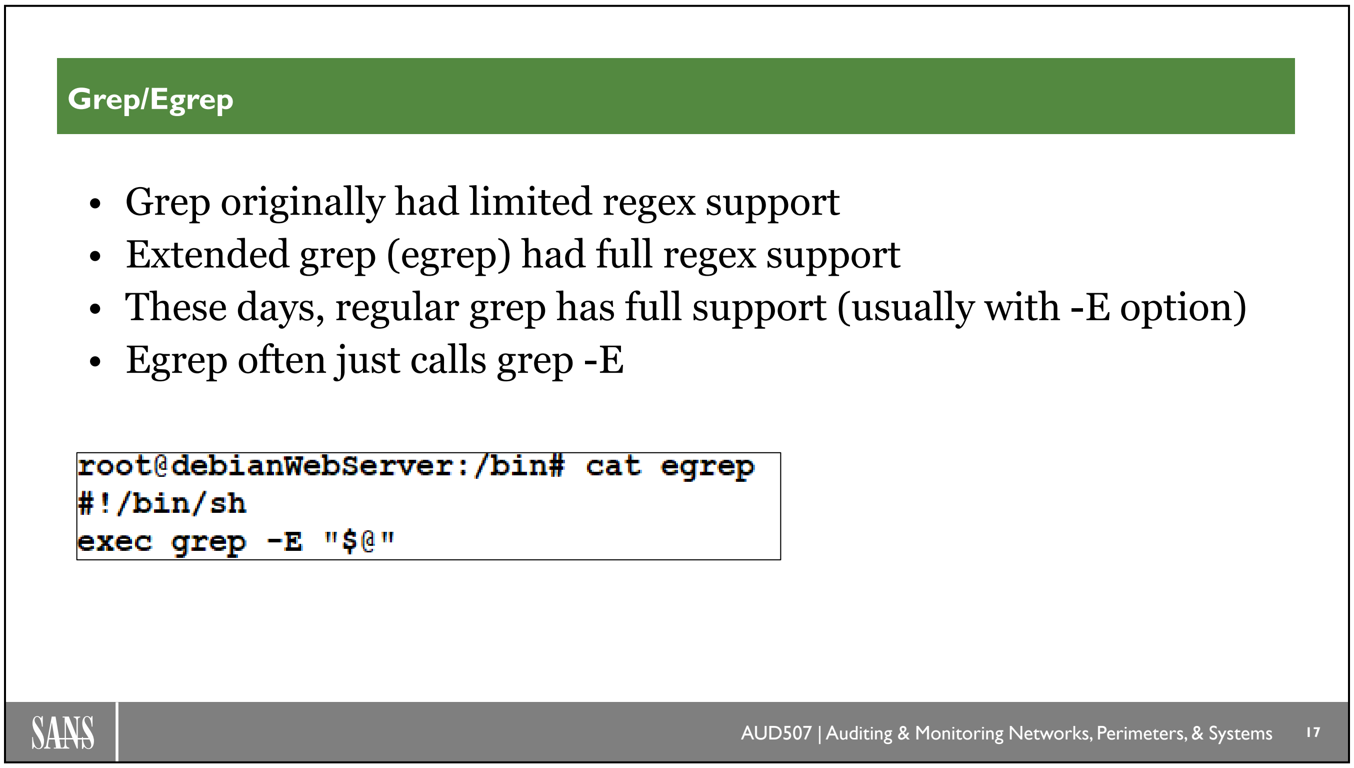
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Grep/Egrep

- Grep originally had limited regex support
- Extended grep (egrep) had full regex support
- These days, regular grep has full support (usually with -E option)
- Egrep often just calls grep -E

```
root@debianWebServer:/bin# cat egrep
#!/bin/sh
exec grep -E "$@"
```

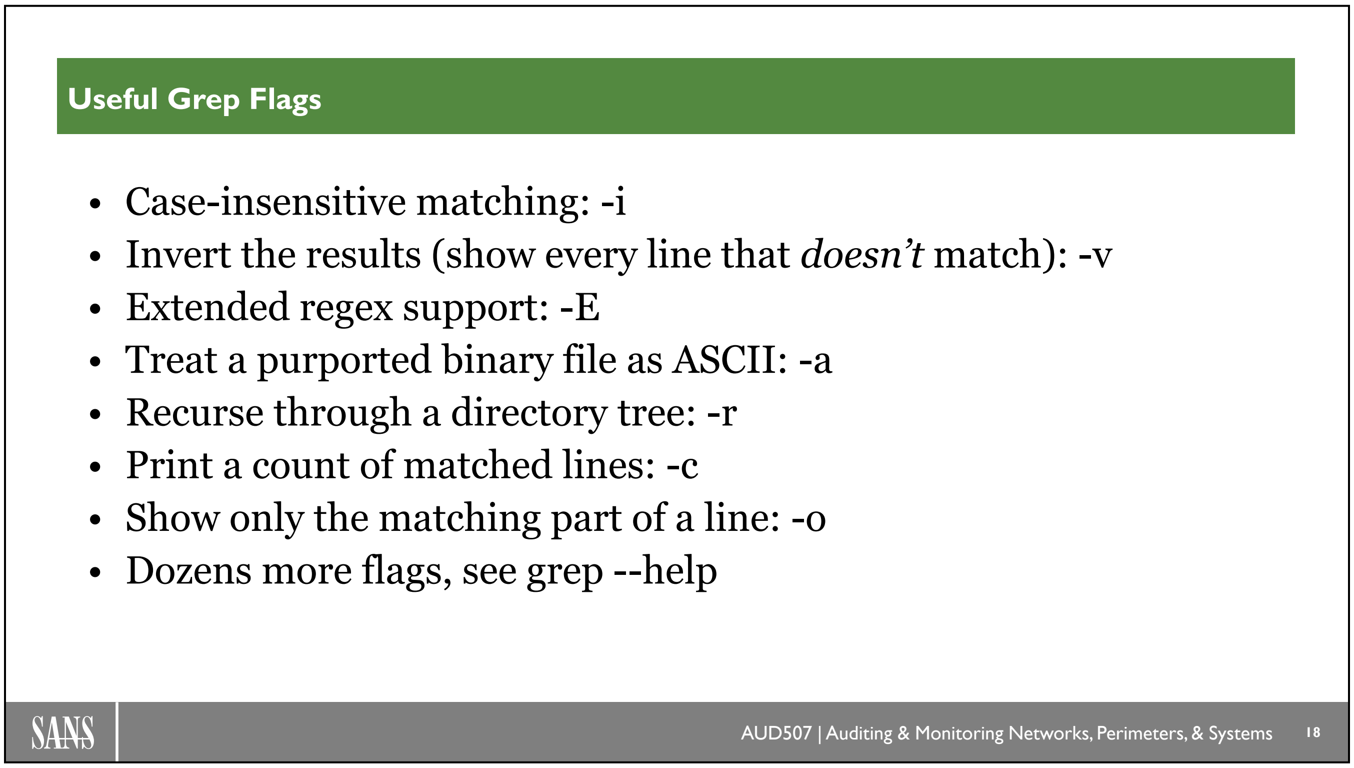
Grep is the original tool, the name of which stands for "Global Regular Expression Print." It has gone through many revisions over the years, for a while being replaced with an extended version called "egrep". Most distributions these days included the extended functionality in the regular grep program, and alias the egrep program to call grep with the flag which tells it to use "extended" regular expressions.

The original grep has a somewhat limited regular expression library that it supports. Actually, it might be more accurate to say that grep supports the original expression set and egrep supports an extended set. Most notable in the egrep library is better support for character classes, character counts, back references, and more. We will look at some of these features as we work our way through the material, working to put together a baseline audit script. Before we dig in any further, however, we should take a few minutes to discuss regular expressions themselves.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Useful Grep Flags

- Case-insensitive matching: -i
- Invert the results (show every line that *doesn't* match): -v
- Extended regex support: -E
- Treat a purported binary file as ASCII: -a
- Recurse through a directory tree: -r
- Print a count of matched lines: -c
- Show only the matching part of a line: -o
- Dozens more flags, see grep --help

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 18

Grep has a number of flags which are useful to the auditor. This slide lists the ones that we commonly use during audit work.

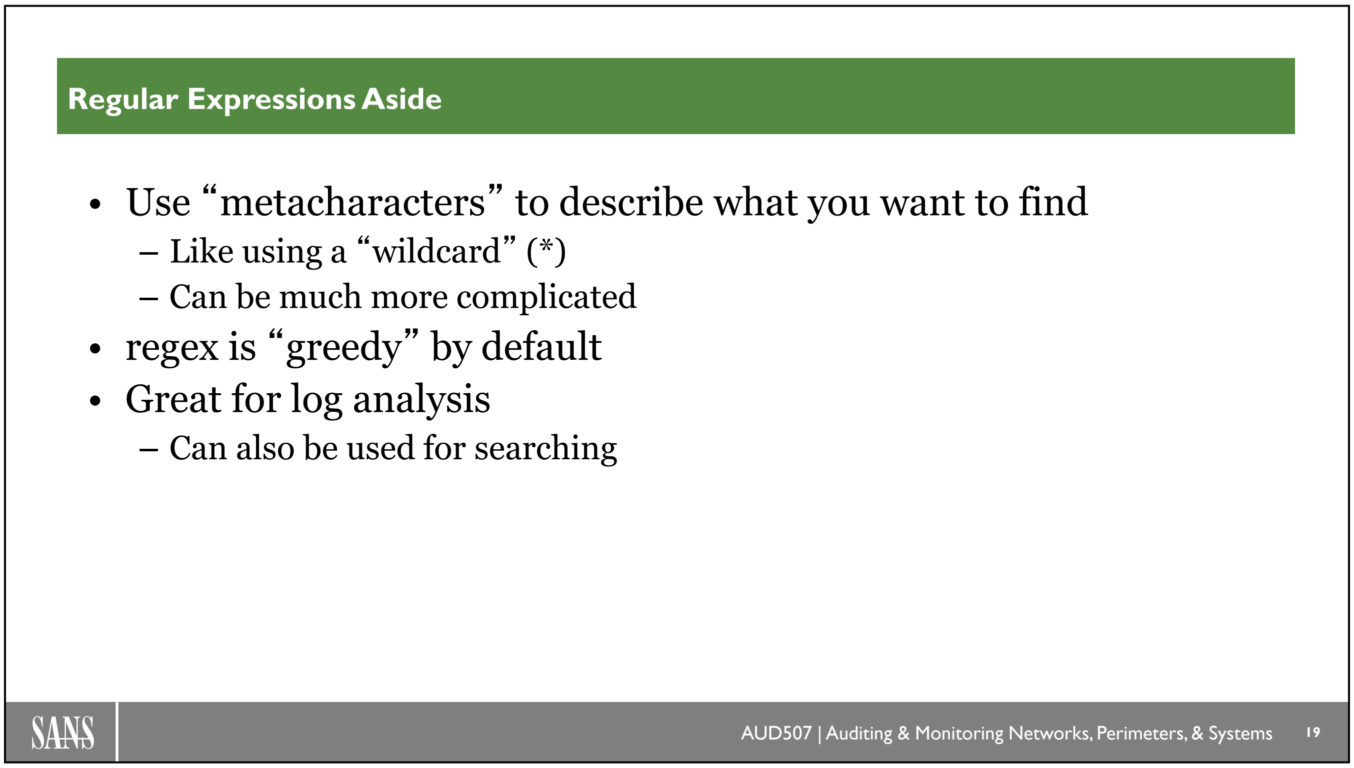
- Case-insensitive matching: -i
- Invert the results (show every line that *doesn't* match): -v
- Extended regex support: -E
- Treat a purported binary file as ASCII: -a
- Recurse through a directory tree: -r
- Print a count of matched lines: -c
- Show only the matching part of a line: -o

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Regular Expressions Aside

- Use "metacharacters" to describe what you want to find
  - Like using a "wildcard" (*)
  - Can be much more complicated
- regex is "greedy" by default
- Great for log analysis
  - Can also be used for searching

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 19

Another important capability within UNIX systems that can be used in connection with the 'find' utility and can also be applied to log analysis and other purposes is regular expression (regex) matching.

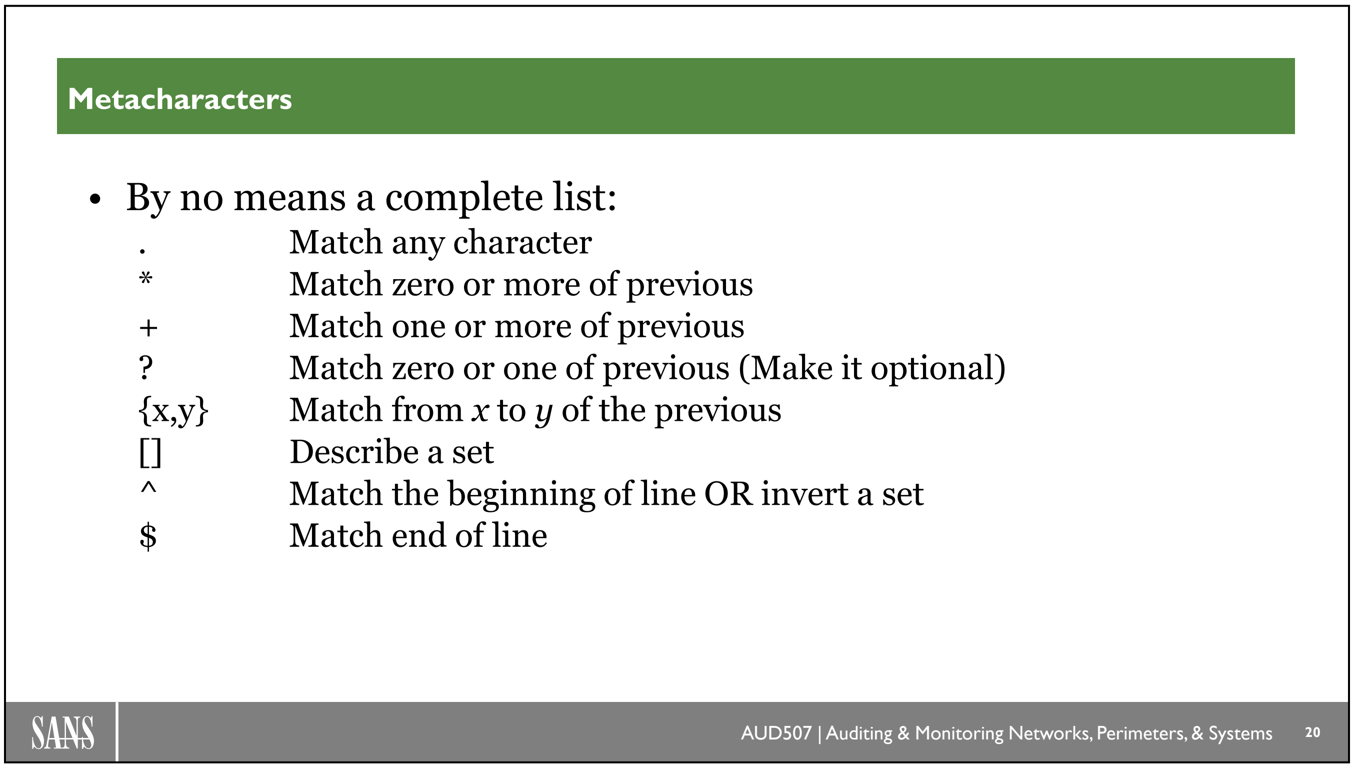
Regular expressions make use of special tokens called "metacharacters" to describe patterns of characters that we would like to extract or identify in a text file or other data source. You can liken this to using the asterisk as a wildcard when getting a directory listing, but these are far more complicated and far more flexible.

An important fact to know about regular expressions is that, by default, they are greedy. What this means is that they will always match as soon as possible and as much as possible. This is important because it might seem that regex should find the best match, but this is simply not the case.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Metacharacters

- By no means a complete list:

.	Match any character
*	Match zero or more of previous
+	Match one or more of previous
?	Match zero or one of previous (Make it optional)
{x,y}	Match from *x* to *y* of the previous
[]	Describe a set
^	Match the beginning of line OR invert a set
$	Match end of line

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 20

Let's start with these few metacharacters listed in the slide. This is by no means a complete list! There are many metacharacters. Even the characters listed here can change meanings depending on the context in which they are used.

Of great importance is the fact that some of these characters are used to match zero, one, or more of the previous character. This is very important because in other contexts, the asterisk is used to match anything, not just something that matches the last expression.

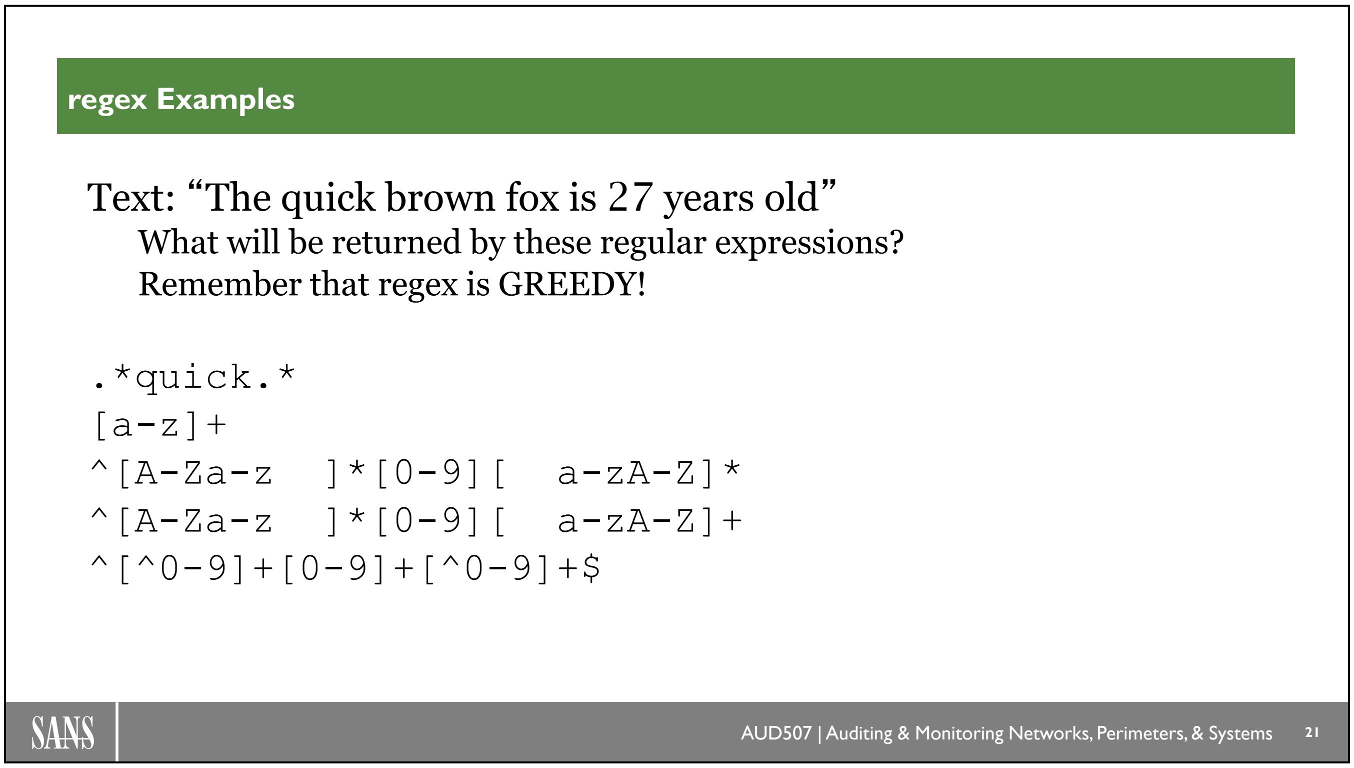
We can also combine these metacharacters. For instance, to match zero or more of any character, the expression would actually be ".*". To match one or more, we would use ".+". Over the next few slides, the instructor will help the class to appreciate what the regular expression matches in the examples given. If you are working with this material at home, the very last slide in the book has the answers. No cheating!

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



regex Examples

Text: “The quick brown fox is 27 years old”

What will be returned by these regular expressions?

Remember that regex is GREEDY!

```
.*quick.*
[a-z]+
^[A-Za-z  ]*[0-9][  a-zA-Z]*
^[A-Za-z  ]*[0-9][  a-zA-Z]+
^[^0-9]+[0-9]+[^0-9]+$
```

SANS

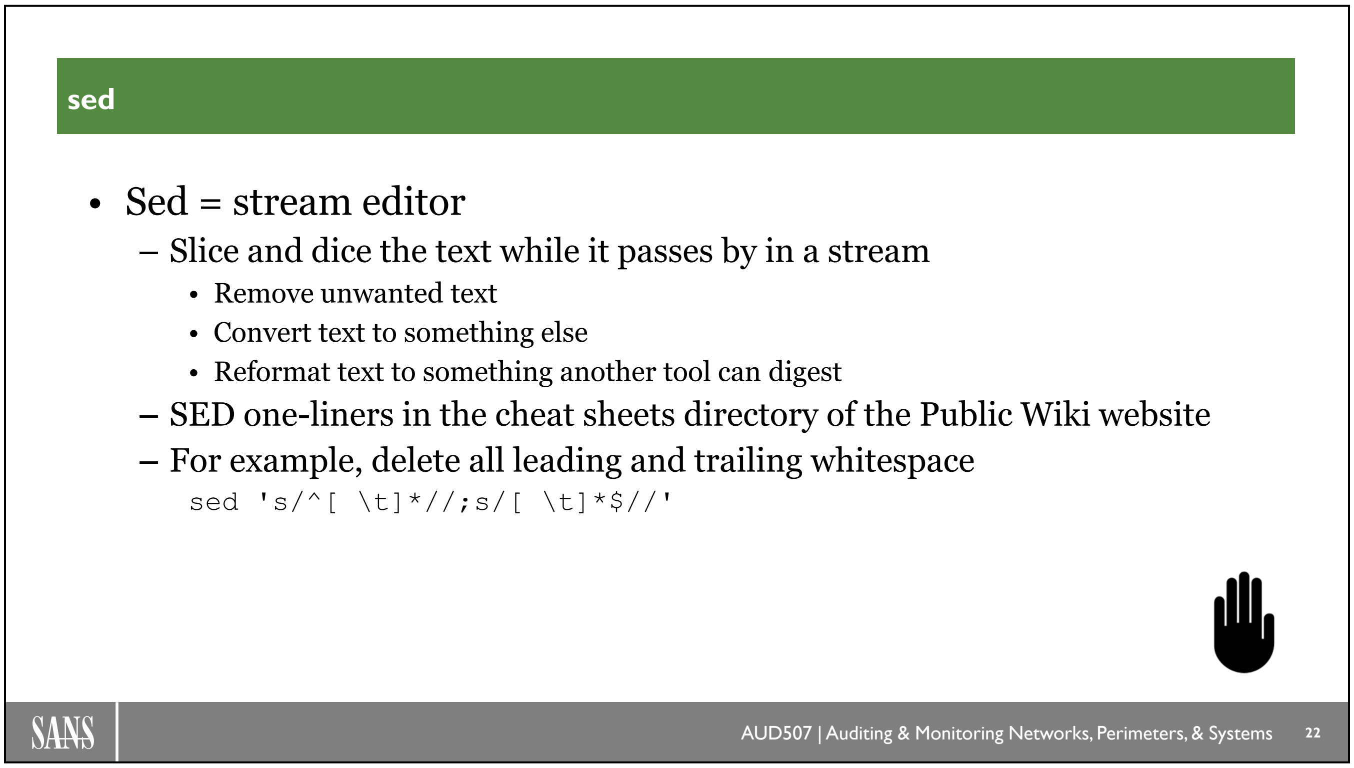
AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 21

Your instructor will work through several example with you to reinforce how regular expressions work. Consider a text file containing the text "The quick brown fox is 27 years old” – given each of the regular expressions on the slide, what text, if any, do you think would be returned?

August 10, 2021

August 10, 2021

© SANS Institute 2021



sed

- Sed = stream editor
  - Slice and dice the text while it passes by in a stream
    - Remove unwanted text
    - Convert text to something else
    - Reformat text to something another tool can digest
  - SED one-liners in the cheat sheets directory of the Public Wiki website
  - For example, delete all leading and trailing whitespace

```
sed 's/^[ \t]*//;s/[ \t]*$//'
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 22

Our next tool is sed. Sed is the stream editor. It allows you to dynamically perform editing operations on streaming data one line at a time. Sed also makes use of regular expressions, both for matching the text that interests you and for replacing or editing that text if that is your desire.

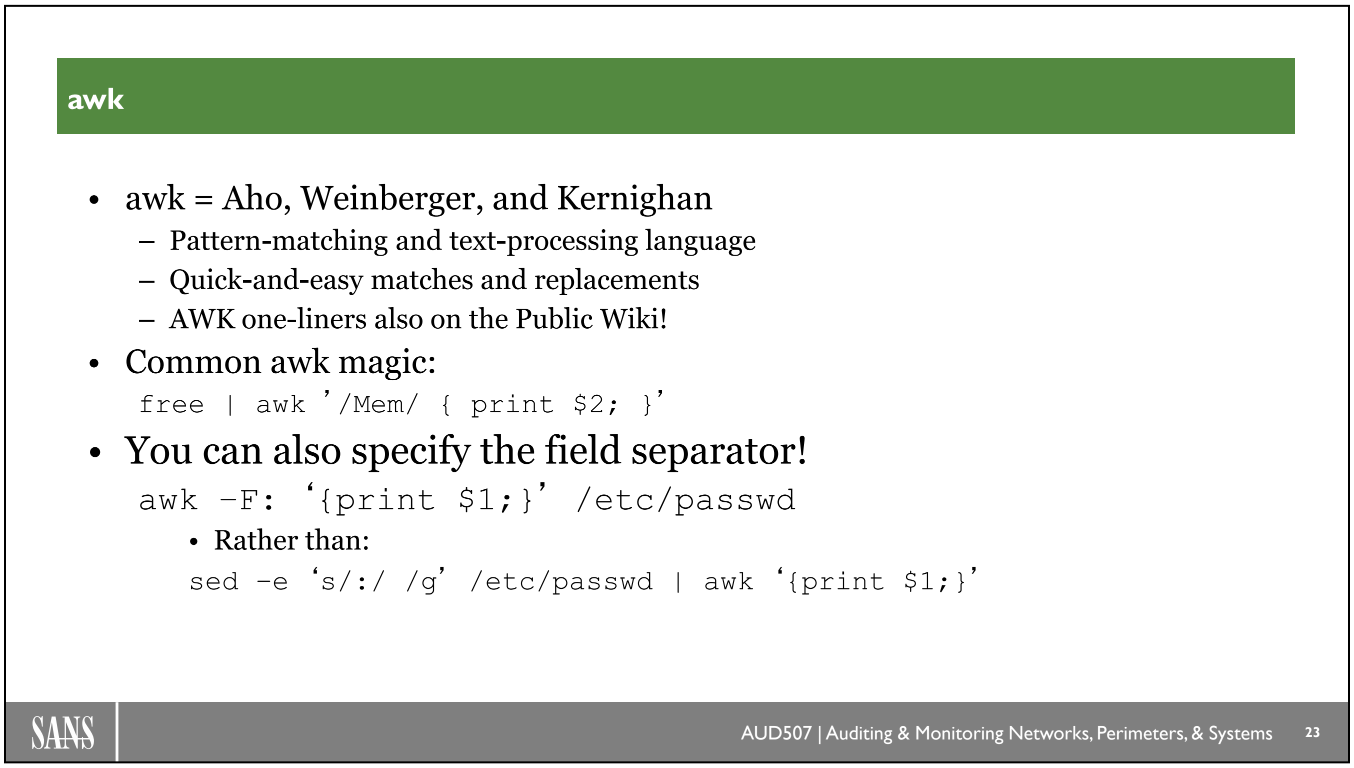
One of the most common uses for sed is to reformat text into the format expected by the next tool in the pipeline. For instance, if you need to append a domain name to a list of host names, sed can do this for you pretty easily.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



awk

- awk = Aho, Weinberger, and Kernighan
  - Pattern-matching and text-processing language
  - Quick-and-easy matches and replacements
  - AWK one-liners also on the Public Wiki!
- Common awk magic:

```
free | awk '/Mem/ { print $2; }'
```

- You can also specify the field separator!

```
awk -F: '{print $1;}' /etc/passwd
```

  - Rather than:

```
sed -e 's/:/ /g' /etc/passwd | awk '{print $1;}'
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 23

The last tool on our short list of tools to know is awk. Awk doesn't stand for anything more than the initials of the names of the creators of the pattern-matching language that it represents.

Although each tool has its specific uses, awk is more of a general-purpose tool in that it can reproduce the behavior of several of the tools already mentioned. Actually, sed and awk are usually interchangeable, though it is best to be somewhat familiar with both. Although you can solve any text-formatting problem with either one, it is quite likely that a super-complex sed expression will have a relatively simple awk counterpart, and vice versa.

Another useful feature of awk is the ability to specify a field separator. Don't scoff! This can save you a lot of work. What this means is that rather than looking for whitespace, we can tell awk to use something else to split the words on a line. Without this ability, we'd be forced into some unpleasant sed-ness:

```
sed -e 's/:/ /g' /etc/passwd | awk '{print $1;}'
```

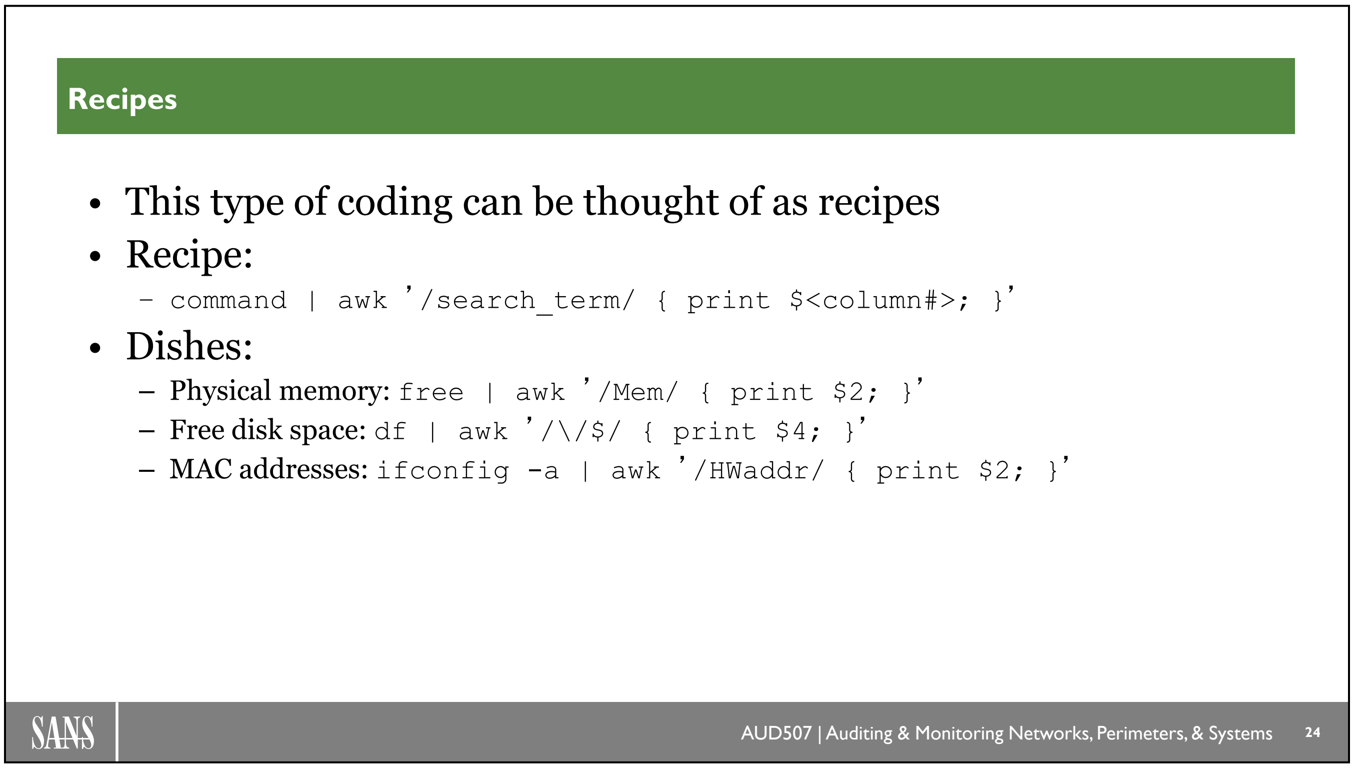
This will accomplish (with a regular expression and sed) the same thing as the example in the slide.

As with the sed tool, we have included the well-known "AWK One Liners" text on the Public Wiki. Most common awk tasks can be found in this file.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Recipes

- This type of coding can be thought of as recipes
- Recipe:
  - `command | awk '/search_term/ { print $<column#>; }'`
- Dishes:
  - Physical memory: `free | awk '/Mem/ { print $2; }'`
  - Free disk space: `df | awk '/\/$/ { print $4; }'`
  - MAC addresses: `ifconfig -a | awk '/HWaddr/ { print $2; }'`

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 24

We'd strongly encourage you to take a step back and think about this in terms of "recipes" rather than focusing too hard on learning to write code. This type of scripting really doesn't require any advanced programming techniques or concepts. All that's really necessary are a few basics and a couple of interesting recipes.

For example, in the slide, we list a recipe that can be used with the AWK tool to extract any column off any line in the output of any command. By generalizing out to a recipe like this, we can create a quick reference chart for ourselves and use that to create our script rather than having to learn all of the ins and outs of AWK.

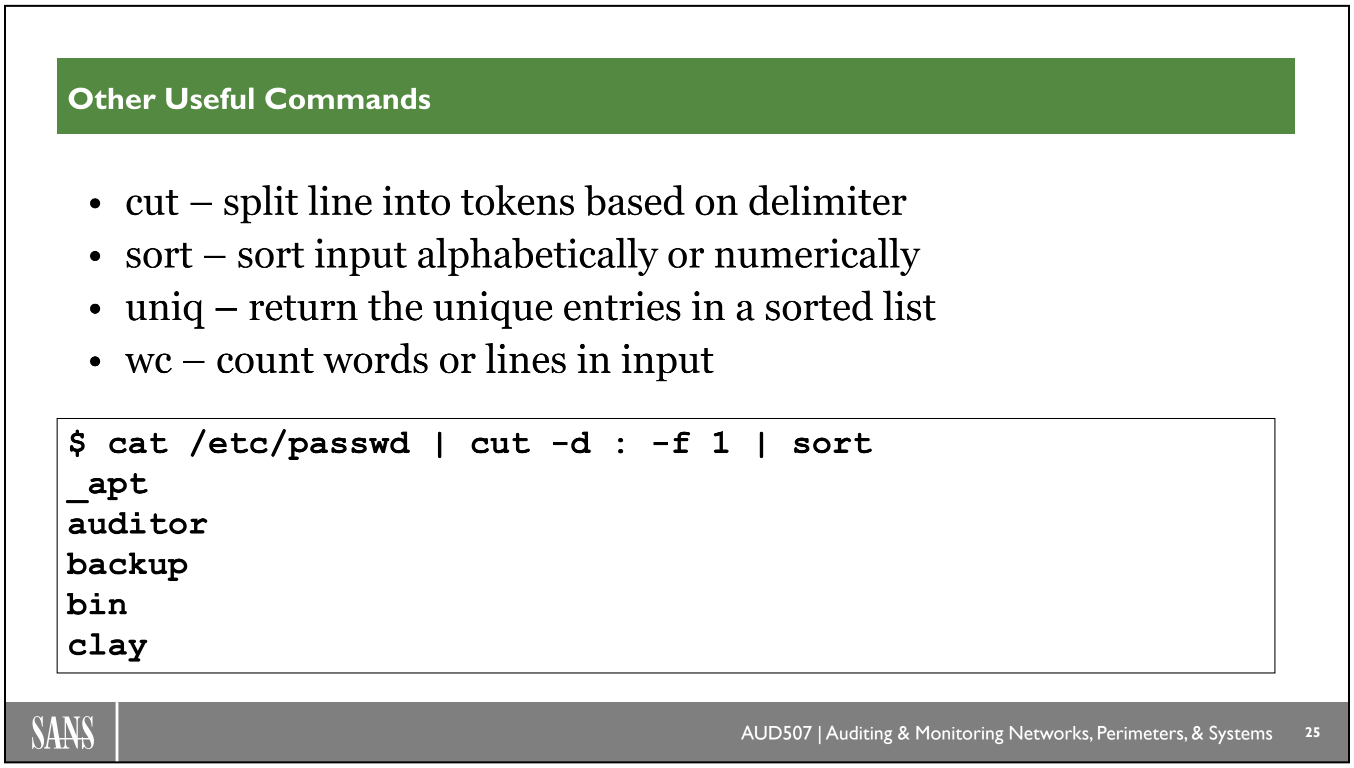
The result is shown in the last three examples. Using that very simple recipe, we can extract out the amount of physical memory installed, the amount of disk space free, and the MAC addresses of all network adapters installed in the system. Of course, we can extract just about anything else that you can think of too!

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Other Useful Commands

- cut – split line into tokens based on delimiter
- sort – sort input alphabetically or numerically
- uniq – return the unique entries in a sorted list
- wc – count words or lines in input

```
$ cat /etc/passwd | cut -d : -f 1 | sort
_apt
auditor
backup
bin
clay
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 25

There are several other text-manipulation tools which are useful for auditors. We mention a few of them here.

Cut is similar to awk, in that it can print certain parts of a string based on a delimiter.

The sort command will sort text which is passed into it. This is useful for finding and eliminating redundant data (used in conjunction with the uniq command). It is also useful for finding the largest or smallest values in a set of data.

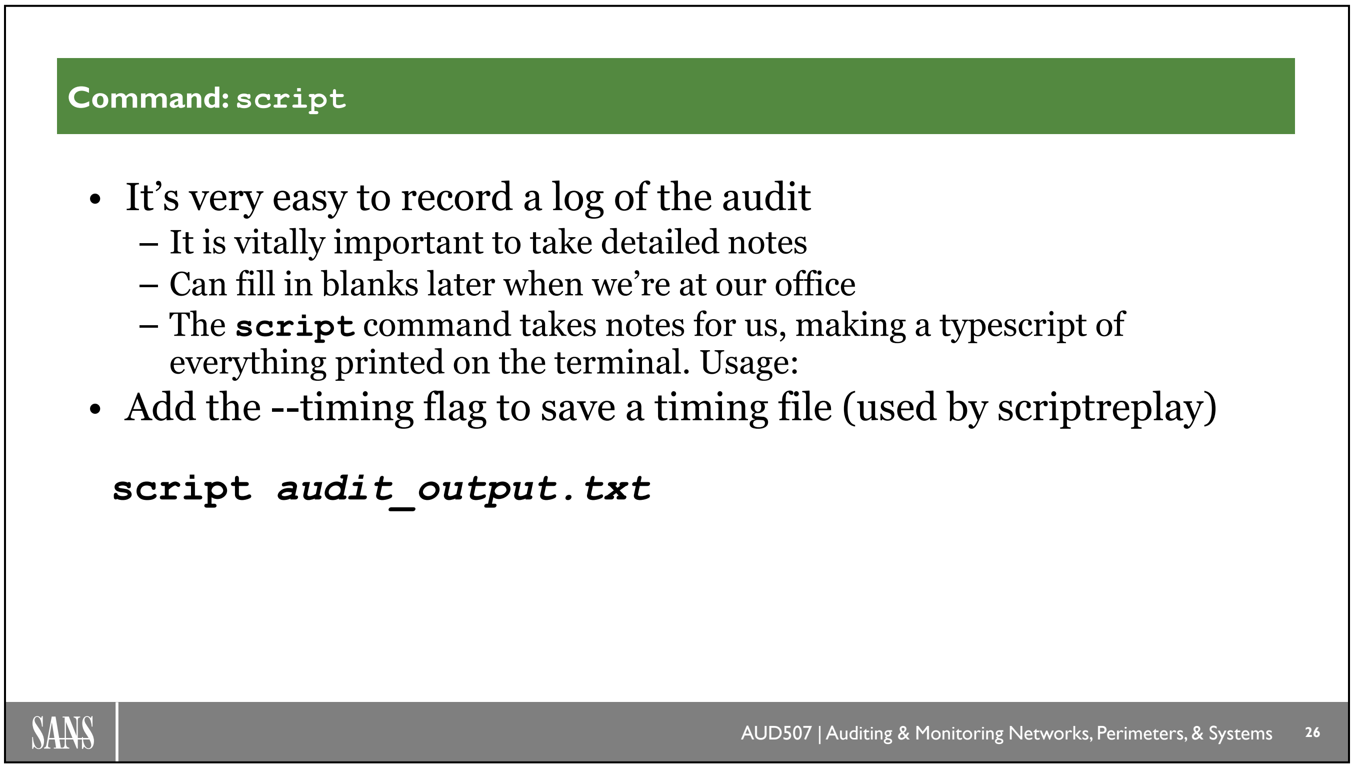
The uniq command takes in a sorted list as input and prints each unique item from the list once on its output. This tool can help you to quickly remove duplicates from a set of data.

The wc command performs word and line counts on the data passed to it. This tool can give you a quick idea of how many lines are in a file, or how any results were returned by a command.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Command: `script`

- It's very easy to record a log of the audit
  - It is vitally important to take detailed notes
  - Can fill in blanks later when we're at our office
  - The **`script`** command takes notes for us, making a typescript of everything printed on the terminal. Usage:
- Add the --timing flag to save a timing file (used by scriptreplay)

```
script audit_output.txt
```

When you run the script command, it will begin to record every command that is typed and everything that appears in the terminal window where it's running. When you're finished with all the tests, simply have the administrator type the word "exit" or press Control-D. This will cause the recording to stop and generate a file. Within the file, you will have a starting timestamp, a record of everything that was done, a record of all of the results, and an ending timestamp.

We'd still strongly recommend that the audit always begin with this command; this should definitely become a part of your audit evidence. Not only will it document for you everything that was done, but it can also be a very valuable piece of evidence should someone eventually claim that you did something that you didn't!

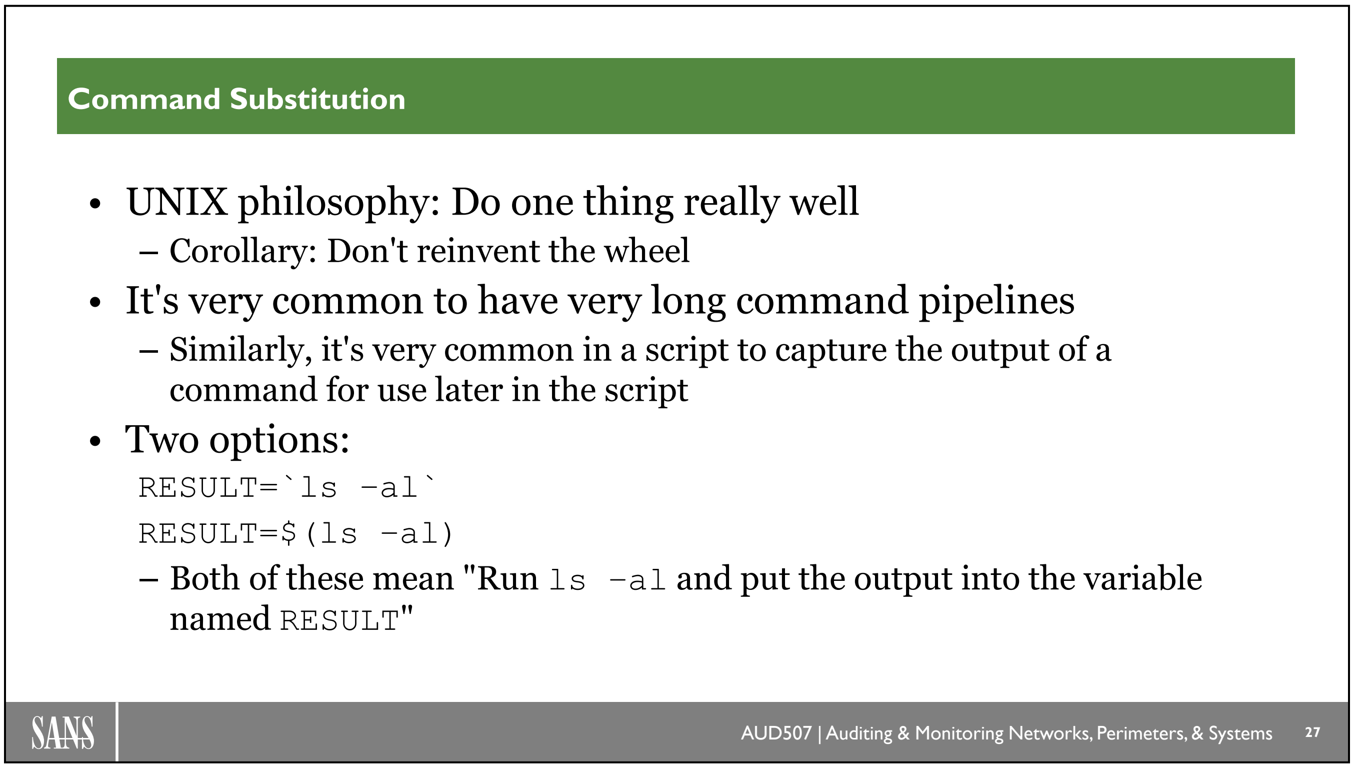
If you choose to add the --timing (or -t) flag with a filename, you can even replay the session back in real-time after the typescript file is saved.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Command Substitution

- UNIX philosophy: Do one thing really well
  - Corollary: Don't reinvent the wheel
- It's very common to have very long command pipelines
  - Similarly, it's very common in a script to capture the output of a command for use later in the script
- Two options:

```
RESULT=`ls -al`
RESULT=$(ls -al)
```

  - Both of these mean "Run `ls -al` and put the output into the variable named `RESULT`"

In the UNIX world, systems programmers generally follow the philosophy that the tools that they write should do one thing really well rather than doing many things. This tends to be a good thing and could be said to follow the "economy of mechanism" security principle. This philosophy, however, tends to lead to four, five, six, or even more commands hooked together into a long command pipeline.

Since these command pipelines are so powerful, we will definitely have them in our scripts. Frequently, we will want to capture the output of one of these pipelines into a variable so that we can act on it or manipulate it in some way. There are two ways to accomplish this.

The first method is using the "backtick" or "backquote." This character is typically located at the top-left corner of your keyboard along with the tilde (~). Consider this:

```
RESULT=`ls -al`
```

August 10, 2021

This tells the shell to execute the ls –al command and put the resulting text into a variable named RESULT. We can then perform any other operation on that text that we desire since it is now simply *text* rather than a *command*.

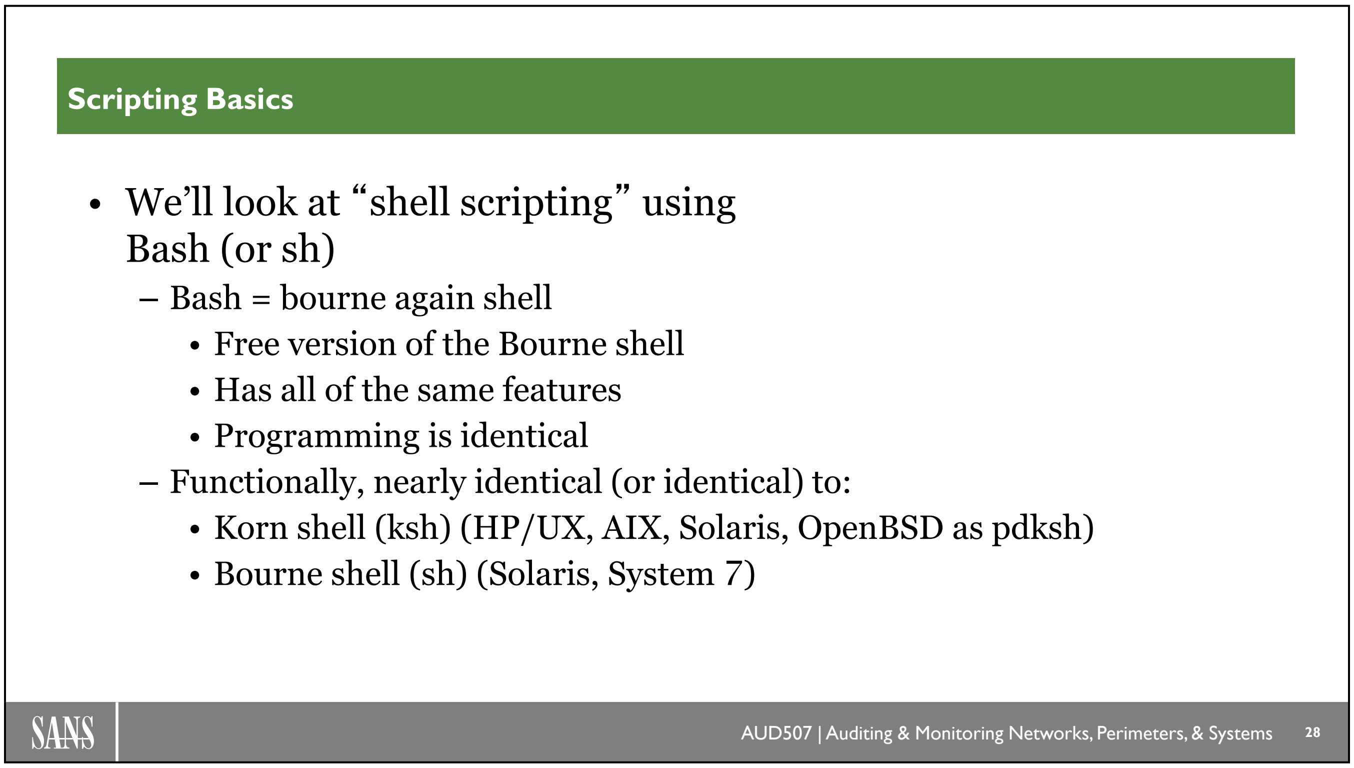
An alternative notation uses this structure:

```
$(<command [| command…]>)
```

For the sake of easy reading, the `$()` notation is easier to see and identify. Using backticks is completely legal, and very common, but it can be easy to mistake the backtick for some other character.

August 10, 2021

© SANS Institute 2021



Scripting Basics

- We'll look at "shell scripting" using Bash (or sh)
  - Bash = bourne again shell
    - Free version of the Bourne shell
    - Has all of the same features
    - Programming is identical
  - Functionally, nearly identical (or identical) to:
    - Korn shell (ksh) (HP/UX, AIX, Solaris, OpenBSD as pdksh)
    - Bourne shell (sh) (Solaris, System 7)

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 28

The Bash shell, or "bourne again shell," is a free (GPL) version of the Bourne shell. Many of the shells that you will use on a UNIX system support all of the basics that we will cover this afternoon. This means that you can take everything we talk about here and any scripts we write and use them on most other UNIX-based systems with little or no modification.

In fact, the reason we have selected Bash as our example is that the shells listed in the slide are scripted using precisely the same language. While the invocation of the shell may change, the various scripting commands covered here, and in the lab, will be the same from system to system. This gives us a useful one-size-fits-all tool.

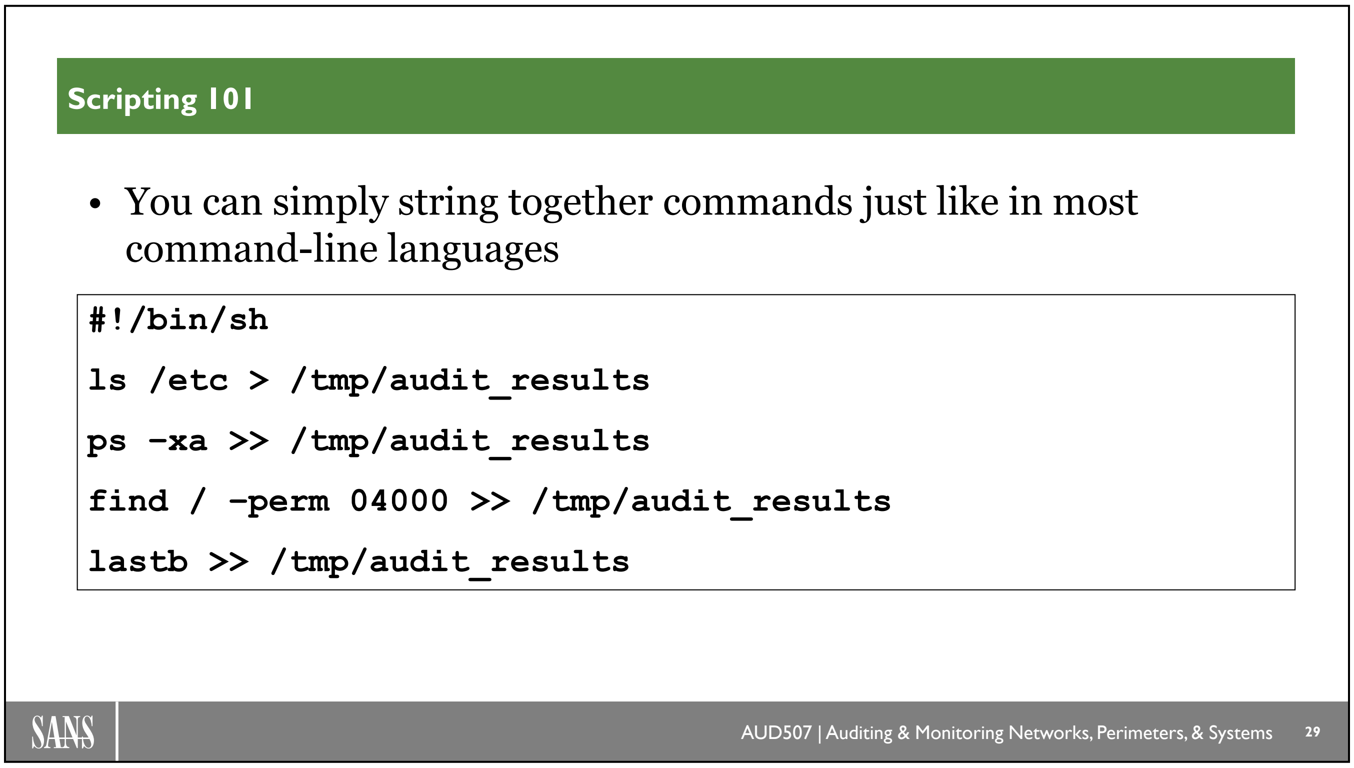
Even if the features or syntax are slightly different, you can always use the online manual (man) to look up the details.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Scripting 101

- You can simply string together commands just like in most command-line languages

```
#!/bin/sh
ls /etc > /tmp/audit_results
ps -xa >> /tmp/audit_results
find / -perm 04000 >> /tmp/audit_results
lastb >> /tmp/audit_results
```

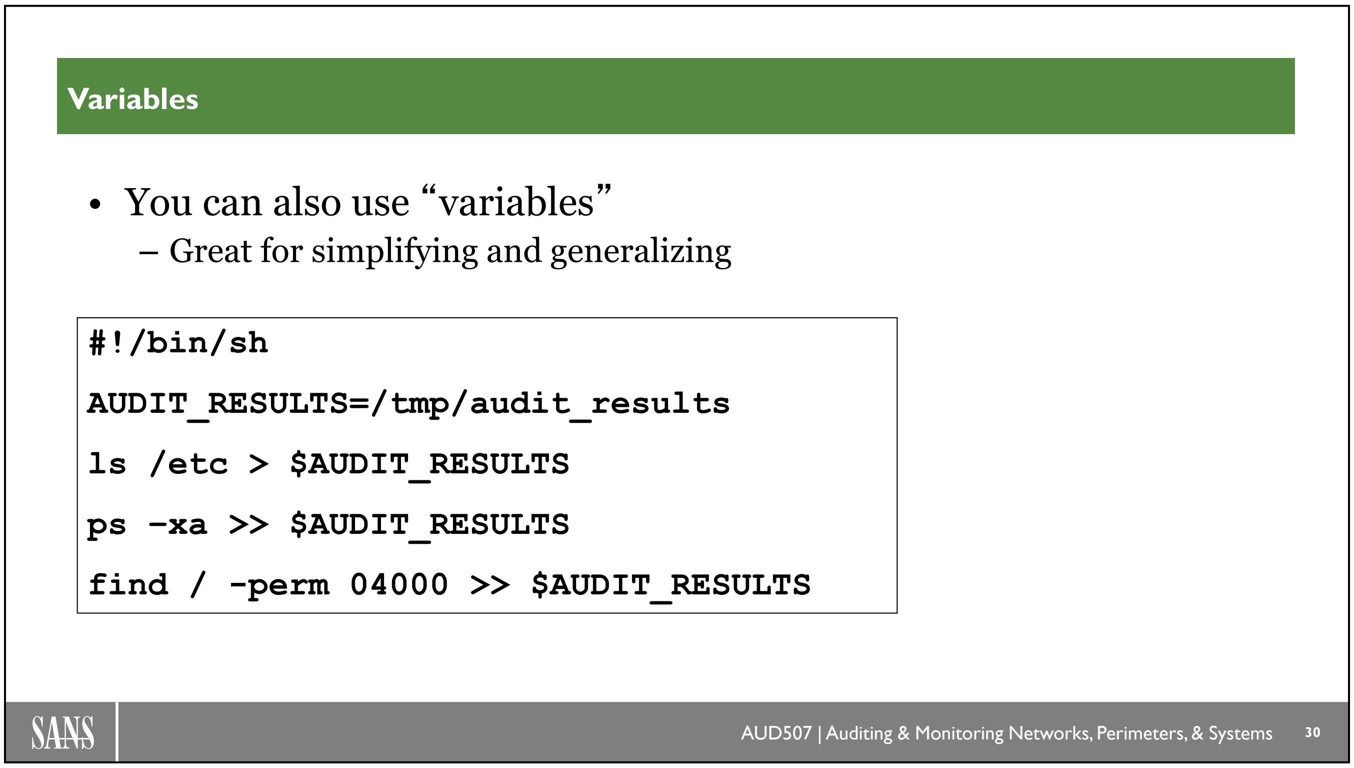
UNIX scripting—particularly Bash, csh, tcsh, sh, ksh, and other command-line interpreter scripting—is very typical of the command-line scripting languages for most platforms. If you have a series of commands that you want to run, simply enter them in, line by line, and save them into a file. In fact, this is why scripting is called "scripting"! We're simply writing a "script" that the computer should follow, just like an actor in a play.

We can use all of the normal UNIX commands that are available, and we can even save our results by using output redirection (the > sign). It is very important to begin the script with the phrase "#!/bin/sh". This specifies the path to the shell that should be used to interpret the script. We could also use "#!/bin/bash" in this case, but bash and sh are typically equivalent these days.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Variables

- You can also use “variables”
  - Great for simplifying and generalizing

```
#!/bin/sh
AUDIT_RESULTS=/tmp/audit_results
ls /etc > $AUDIT_RESULTS
ps -xa >> $AUDIT_RESULTS
find / -perm 04000 >> $AUDIT_RESULTS
```

SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 30

One of the things you will want to be able to do very early on in scripting is to generalize your script or make it easily customizable. If there is something that you do repeatedly in your script—for instance, storing the results into a specific file—then this is a perfect target for creating a variable. Notice that in the slide, we specify a variable called “AUDIT_RESULTS”, storing in it “/tmp/audit_results”. Now, every place that we use that filename in the previous slide, we can replace with $AUDIT_RESULTS.

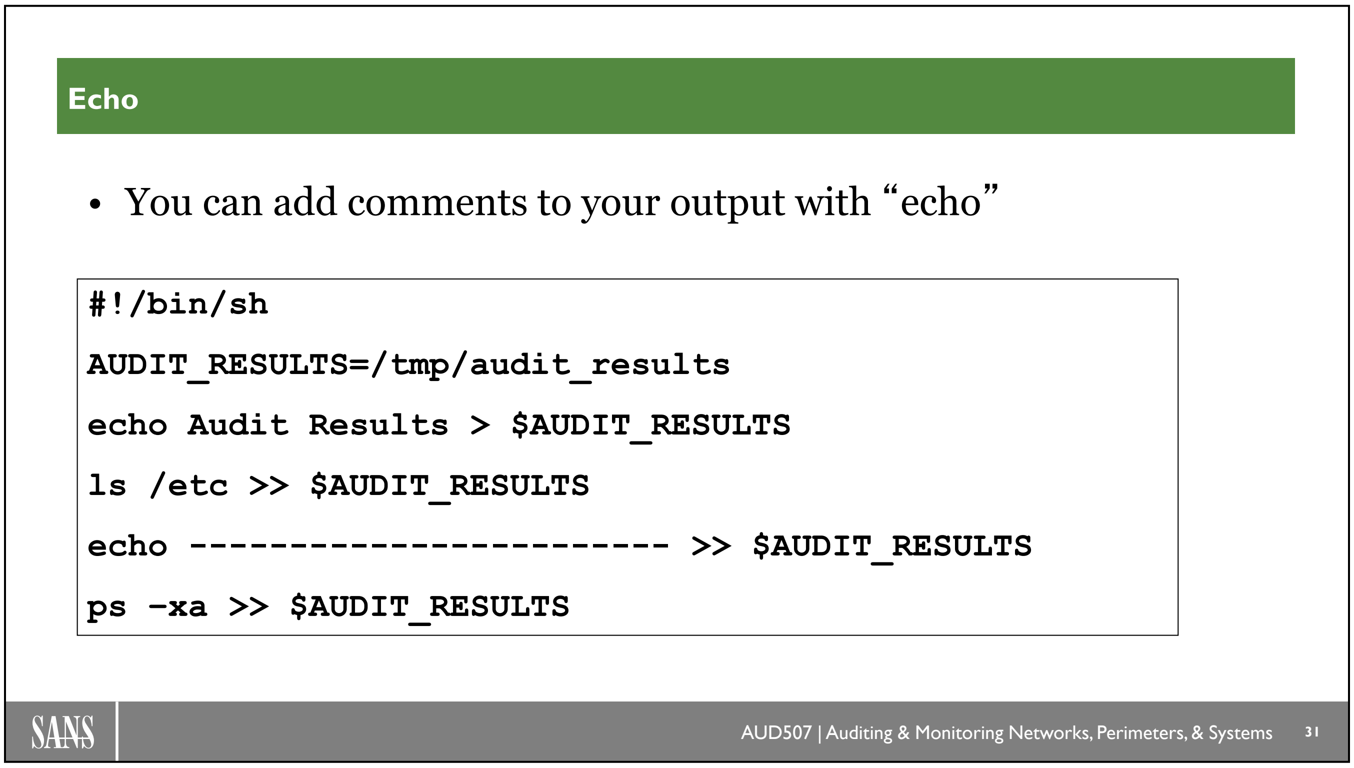
You might be wondering, “What’s the point?” The real key here is that if you want to change the destination for the results, you no longer need to change every single line in the script. One change alters the whole thing! There are some other, more advanced techniques for dynamically generating the contents of a variable. Depending on the skill of the class, the instructor might choose to cover some additional ways of handling variables.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Echo

- You can add comments to your output with "echo"

```
#!/bin/sh
AUDIT_RESULTS=/tmp/audit_results
echo Audit Results > $AUDIT_RESULTS
ls /etc >> $AUDIT_RESULTS
echo ------------------------ >> $AUDIT_RESULTS
ps -xa >> $AUDIT_RESULTS
```

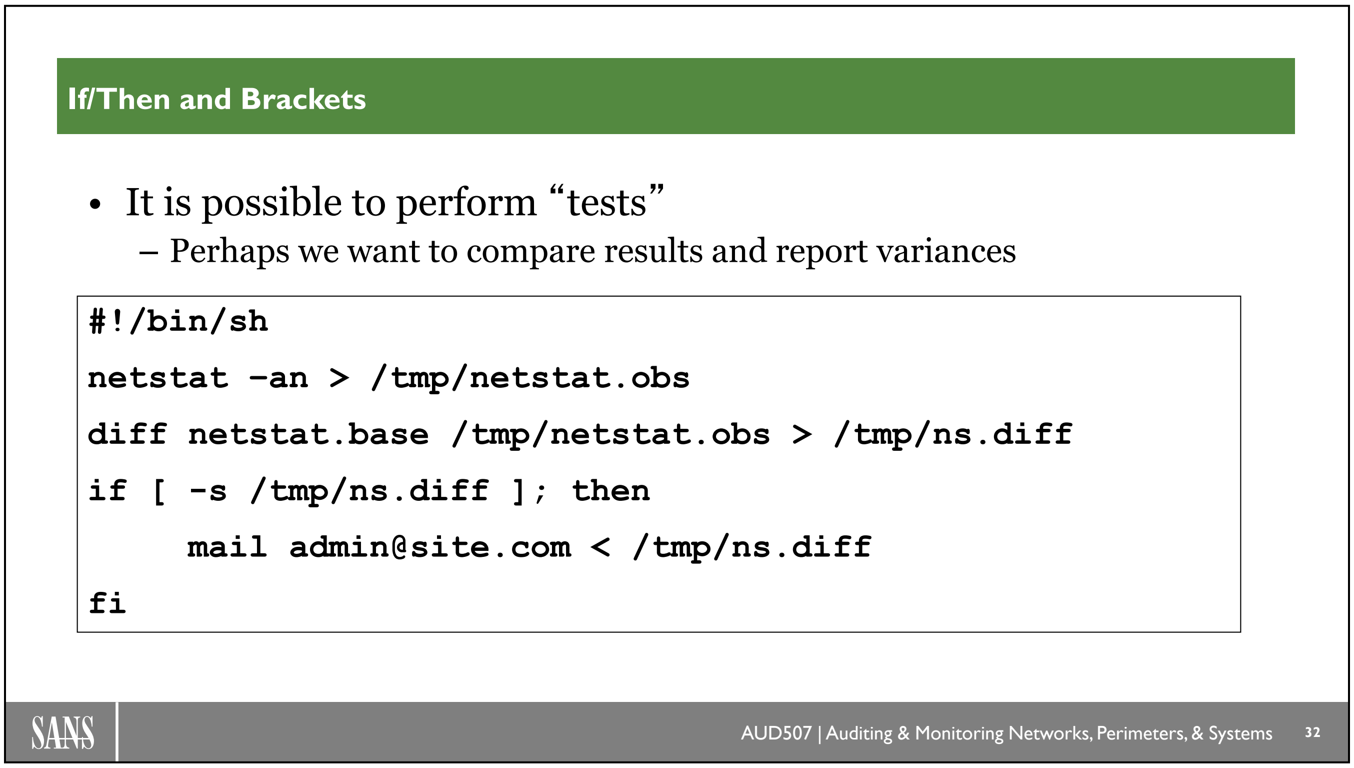
SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 31

So far, our script simply dumps the output of every command into a file. Especially if we're gathering results from many machines, we need a way to distinguish the output from the various systems. To accomplish this, we can use the "echo" command. This command can be used to echo arbitrary strings into our file, too. We can use this to add the name of the host or perhaps the date or any other pertinent piece of information.

August 10, 2021

August 10, 2021

© SANS Institute 2021



If/Then and Brackets

- It is possible to perform “tests”
  - Perhaps we want to compare results and report variances

```
#!/bin/sh
netstat -an > /tmp/netstat.obs
diff netstat.base /tmp/netstat.obs > /tmp/ns.diff
if [ -s /tmp/ns.diff ]; then
     mail admin@site.com < /tmp/ns.diff
fi
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 32

If we’re planning to use the script to perform automated security auditing and reporting, one of our goals is to get the administrators or security officers to look at the reports. If the system continually sends blank or meaningless reports, we will likely not reach our goal, so we might choose to test to see whether or not there were any meaningful results.

In the script shown in the slide, we have previously collected baseline information for listening ports. Now, when the script is running, we generate an observed file. After the observed file is created, the script runs the ‘diff’ command to identify any differences between the two files. To use this to produce our report, we can set up a conditional.

The if/then structure is pictured in the slide and is fairly self-explanatory, but the brackets need some explanation. The square brackets are used to perform tests. In this case, the ‘-s’ is being used to test whether the file size of “ns.diff” is greater than zero. If the file is empty, the script continues. If the file contains anything, the contents are emailed to the administrator at the site.

Notice the spacing! Square brackets should be spaced away from everything. If they aren’t, the shell will try to interpret them as part of an expression, and you will have errors on your hands.

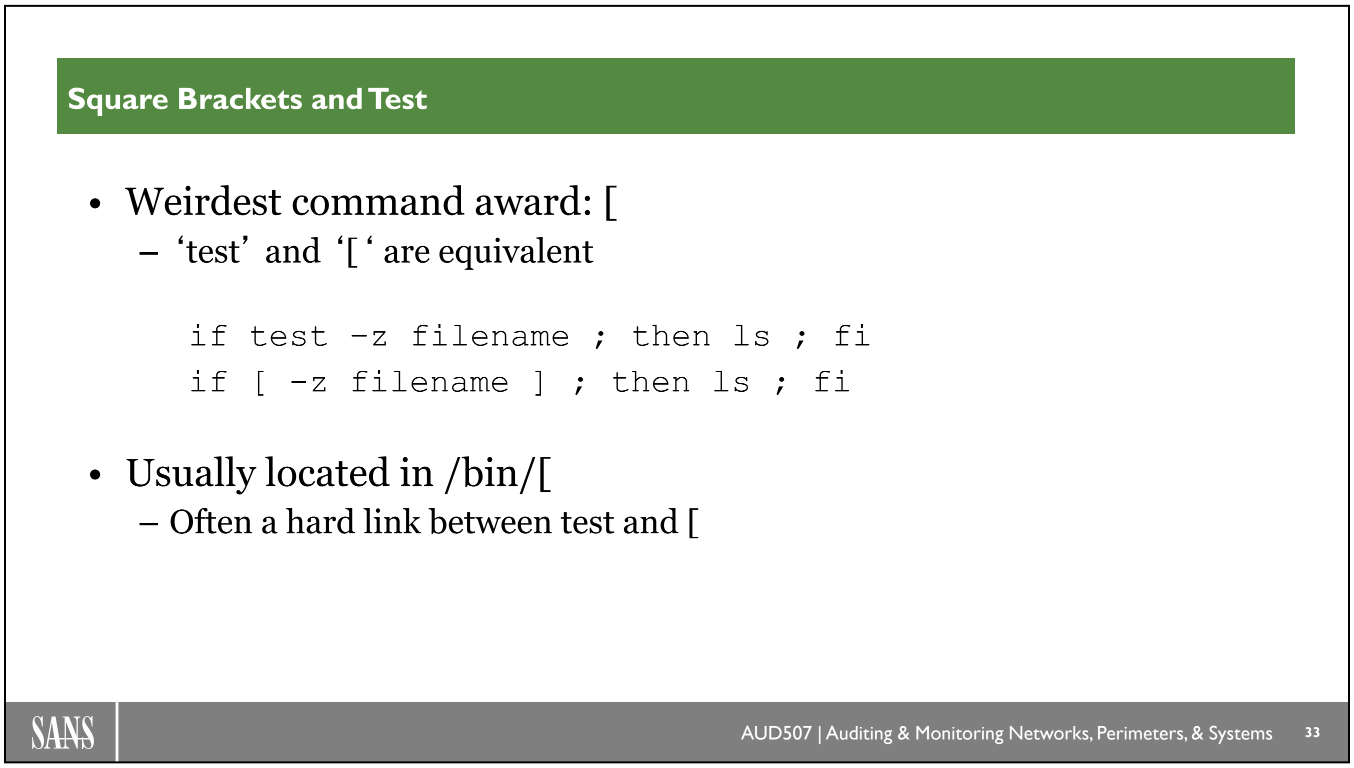
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Square Brackets and Test

- Weirdest command award: [
  - ‘test’ and ‘[‘ are equivalent

```
if test –z filename ; then ls ; fi
if [ -z filename ] ; then ls ; fi
```

- Usually located in /bin/[
  - Often a hard link between test and [

We should explain why the spacing around the brackets is so important. The brackets can be used for set definition ([a-z], for instance) when specifying filenames. If we press the [] up against the ‘if’, then the shell will assume that we are looking for a file named ‘if[….]’ rather than trying to create a conditional expression. The square bracket is actually equivalent to the ‘test’ command. They’re so equivalent that on most systems, you’ll find that they are actually the same file. For instance, notice in this example the inode number:

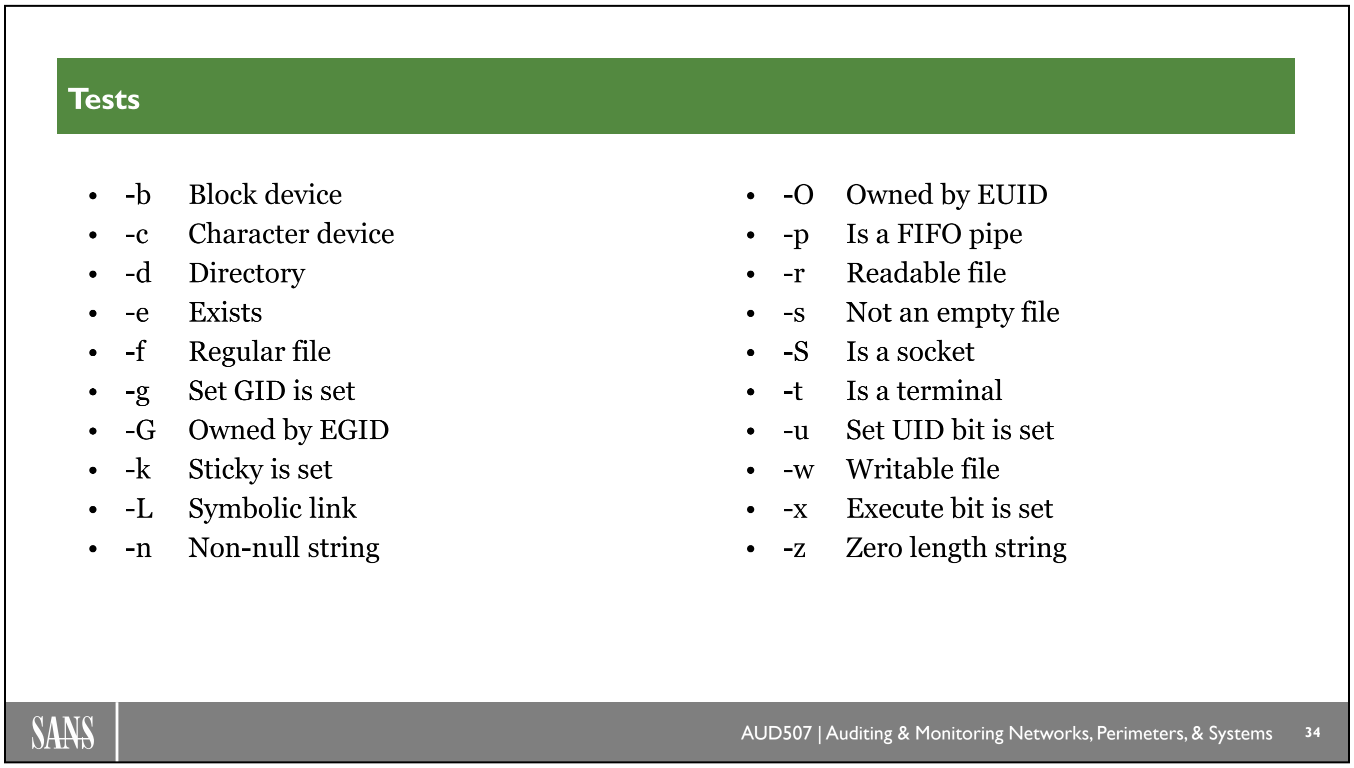
```
david-hoelzers-imac:~ dhoelzer$ ls -li /bin/test /bin/\[
1374818 -r-xr-xr-x  2 root  wheel  46720 Nov 28  2015 /bin/[
1374818 -r-xr-xr-x  2 root  wheel  46720 Nov 28  2015 /bin/test
david-hoelzers-imac:~ dhoelzer$
```

The first number listed on the line in the directory listing is the inode number for the files. The inode is, in a sense, the file pointer. The filename is just an abstraction to refer to the actual inode, which points to the actual file content. As you can see, the ‘test’ and the ‘[’ filenames are “hard linked,” which means that they are two names for the same file. You could think of them as nicknames for the actual file. As an example, you are who you are no matter what name we use to refer to you. The same is true of the contents of inode 1374818 on the system used in the example.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Tests

-b	Block device
-c	Character device
-d	Directory
-e	Exists
-f	Regular file
-g	Set GID is set
-G	Owned by EGID
-k	Sticky is set
-L	Symbolic link
-n	Non-null string
-O	Owned by EUID
-p	Is a FIFO pipe
-r	Readable file
-s	Not an empty file
-S	Is a socket
-t	Is a terminal
-u	Set UID bit is set
-w	Writable file
-x	Execute bit is set
-z	Zero length string

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 34

This testing tool can be used to test for just about any condition you can think of with regard to a file or directory. We can use it to test the type of file (directory, socket, block device, character, and terminal), examine the file permissions relative to the user executing the script (read, write, execute, owner, and group), test to see whether the file is empty, see whether a file exists, etc.

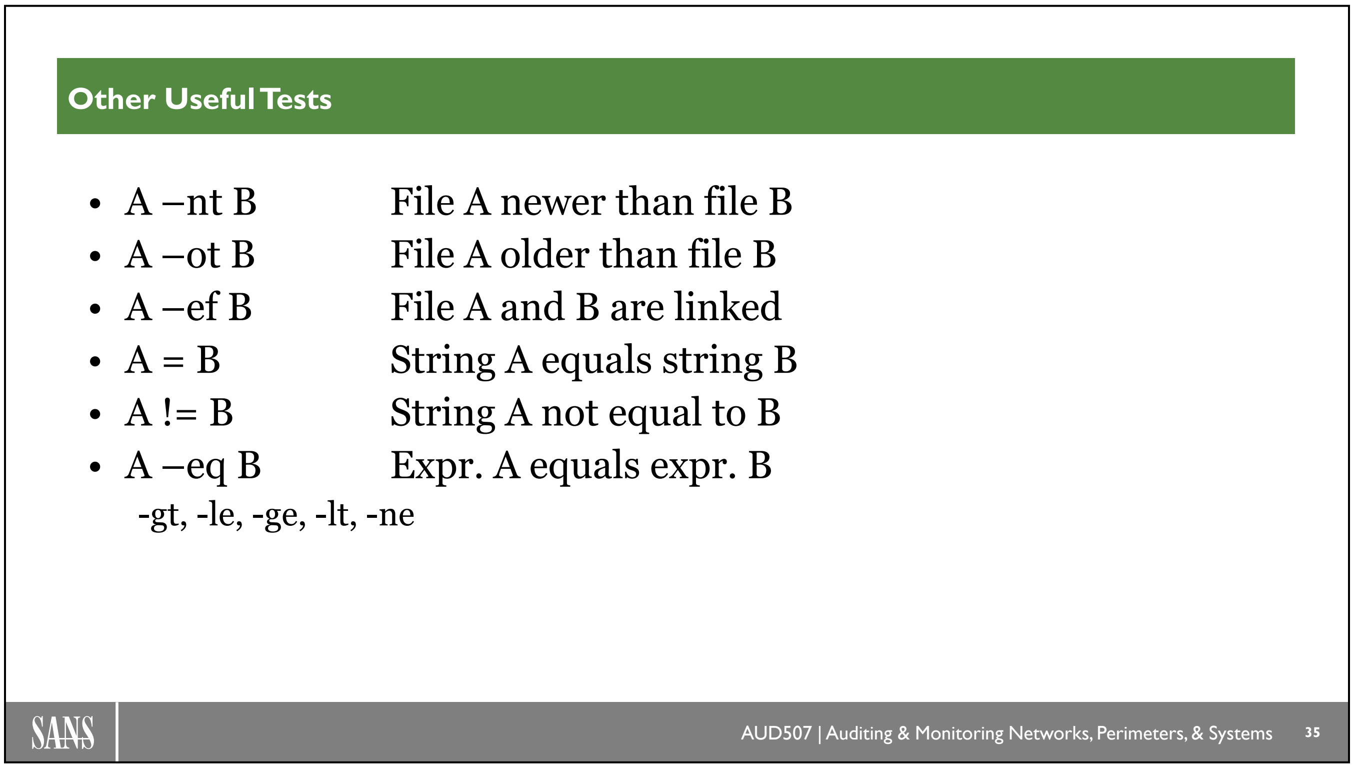
We're not limited to testing files, though that is the primary focus. We can also check the content of strings, looking to see whether they are empty or have contents. Comparisons can be performed as well. We have listed some of the comparisons on the next slide for you.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Other Useful Tests

• A –nt B	File A newer than file B
• A –ot B	File A older than file B
• A –ef B	File A and B are linked
• A = B	String A equals string B
• A != B	String A not equal to B
• A –eq B	Expr. A equals expr. B

-gt, -le, -ge, -lt, -ne

The ability to see whether files are empty, exist, and so on, is very useful, but sometimes we want to perform a comparison. For instance, using the conditionals listed in the slide, we can test to see whether one file is older or newer than another, see whether two files refer to the same file (like test and []), and so on.

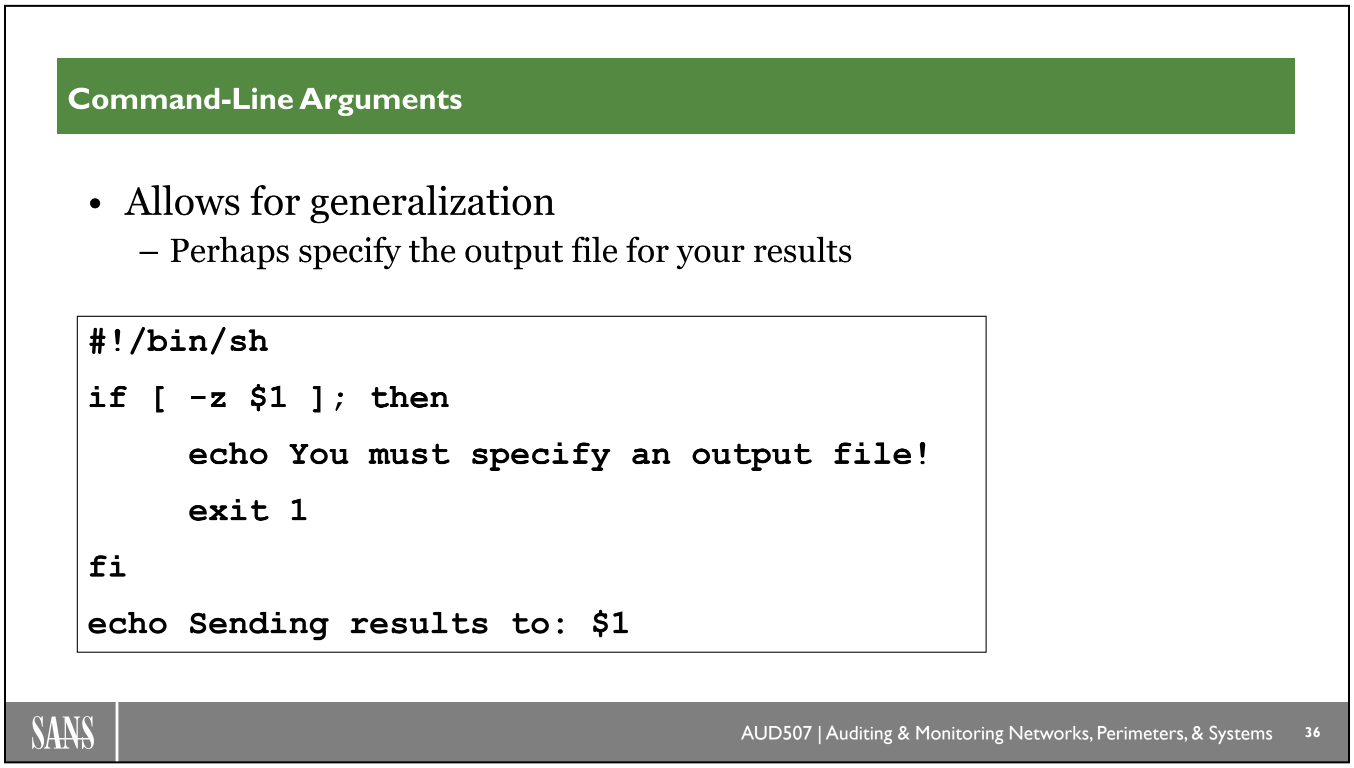
Again, although the primary focus is on files, we also have the ability to use conditionals on strings. These are very straightforward. Strings are either equal or not equal. On the other hand, we might want to perform some type of numerical expression and act on the results. Let's say we wanted to see which of two directories had more files in it. We could do something like this:

```
ONE=`ls / | wc -l`
TWO=`ls /etc | wc -l`
if [ $ONE -gt $TWO ] ; then
            echo Directory one contains more files.
            exit 1
fi
echo Directory two contains more files.
exit 2
```

August 10, 2021

August 10, 2021

© SANS Institute 2021



Command-Line Arguments

- Allows for generalization
  - Perhaps specify the output file for your results

```
#!/bin/sh
if [ -z $1 ]; then
     echo You must specify an output file!
     exit 1
fi
echo Sending results to: $1
```

SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 36

Previously, we looked at using variables to customize or generalize our script. Command-line options give us another possibility for generalization. When you run your script, you can send command-line arguments. For instance:

```
./audit_script /tmp/results
```

This attempts to run the script "audit_script" in the current directory, sending the command-line argument of "/tmp/results". In the script, you can refer to this value using the variable "$1". Notice that we use a different test here, testing to see whether the variable "$1" is empty. For a list of other tests, please refer to the previous two pages for the kinds of tests that can be performed.

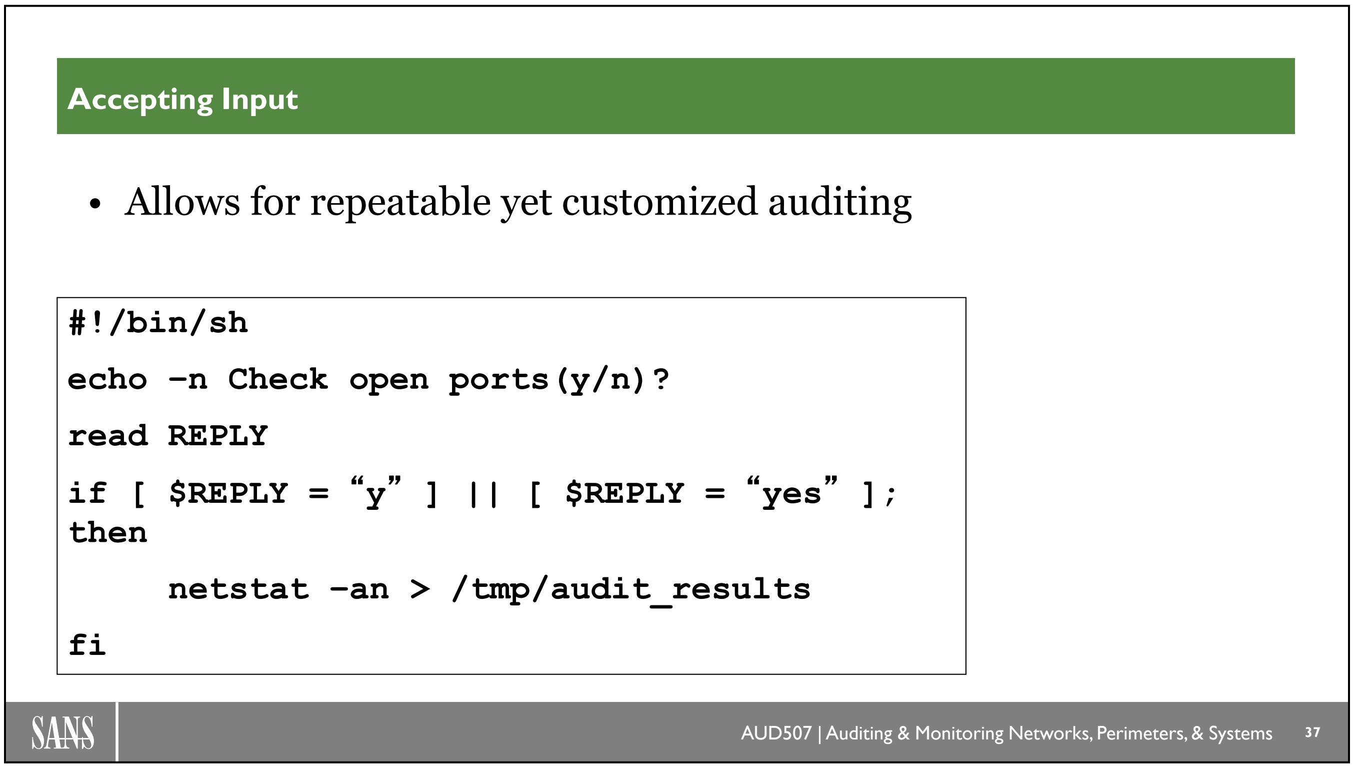
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Accepting Input

- Allows for repeatable yet customized auditing

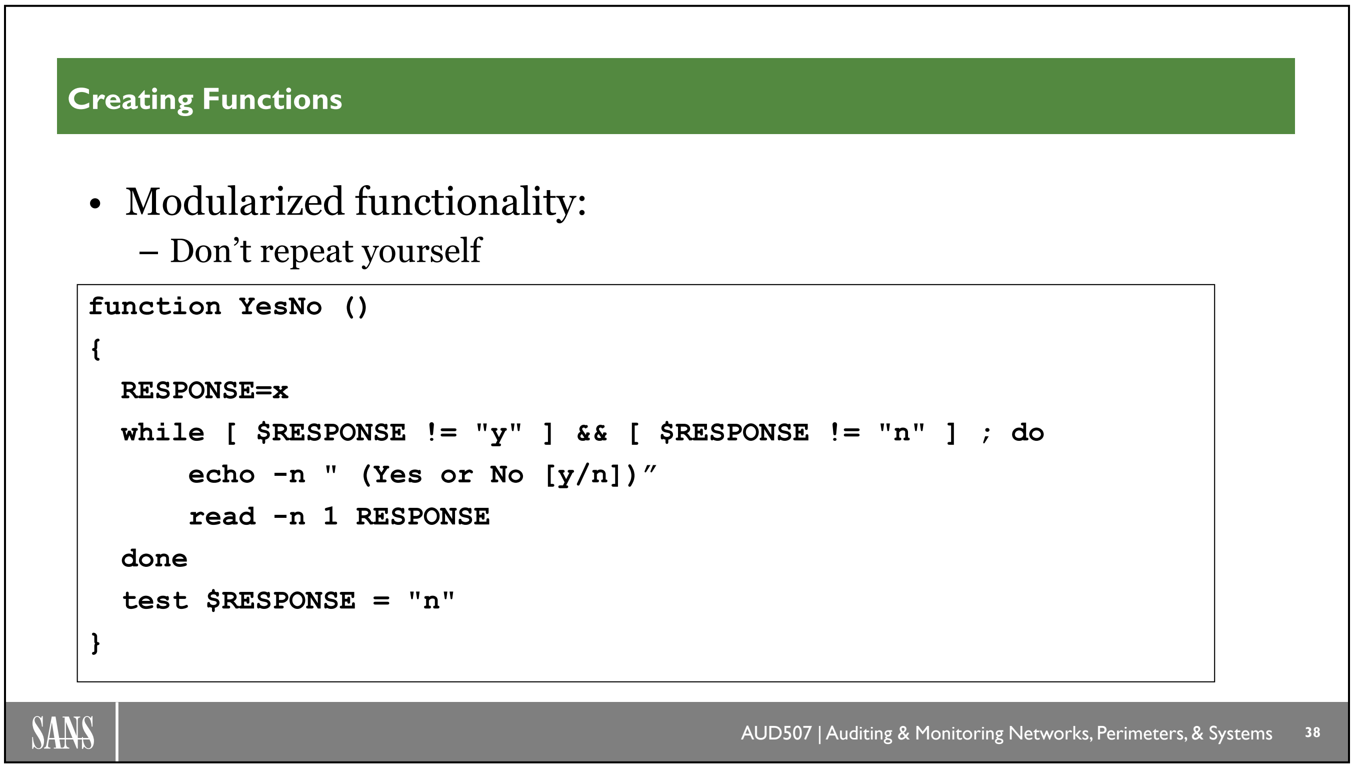
```
#!/bin/sh
echo -n Check open ports(y/n)?
read REPLY
if [ $REPLY = “y” ] || [ $REPLY = “yes” ];
then
     netstat -an > /tmp/audit_results
fi
```

You might also want to take input from the user while the script is running. Perhaps you have written a fairly comprehensive audit script that covers all aspects of a UNIX system. It might also be that your scope is somewhat limited when it comes to the system that you are running the script on. If this is the case, your script can be customized to allow you to choose which options to run on this particular system.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Creating Functions

- Modularized functionality:
  - Don’t repeat yourself

```
function YesNo ()
{
  RESPONSE=x
  while [ $RESPONSE != "y" ] && [ $RESPONSE != "n" ] ; do
      echo -n " (Yes or No [y/n])”
      read -n 1 RESPONSE
  done
  test $RESPONSE = "n"
}
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 38

Probably one of the most important constructs in programming beyond conditionals is functions. A function is simply a collection of actions that you want to abstract out into a single step that can be repeated at any time.

Much like variables, which allow us to define a value in one place and reuse it in many, thereby allowing us to centralize changes to values, functions allow us to define actions in one place and reuse them in many places. Now the location of the *functionality* is centralized, again allowing for easy and quick modification.

The really good coding mantra to follow is “Don’t repeat yourself!” If you find yourself performing the same activity a second time and that action is more than just a line or two of code, you should seriously think about how that functionality can be modularized.

The example on the slide abstracts out a “Yes/No” function. The second-to-last line in the script does deserve special mention, however. Here, we have a bare “test” statement, and you might notice that there’s apparently nothing relaying the results back to the code that called this function. The test function, in this case, will set the return code (which can be accessed using the built-in variable $?). If the response is ‘n’, $? is set to be 0. The only other possibility is ‘y’, which will result in $? being set to 1, allowing us to determine what happened in the function.

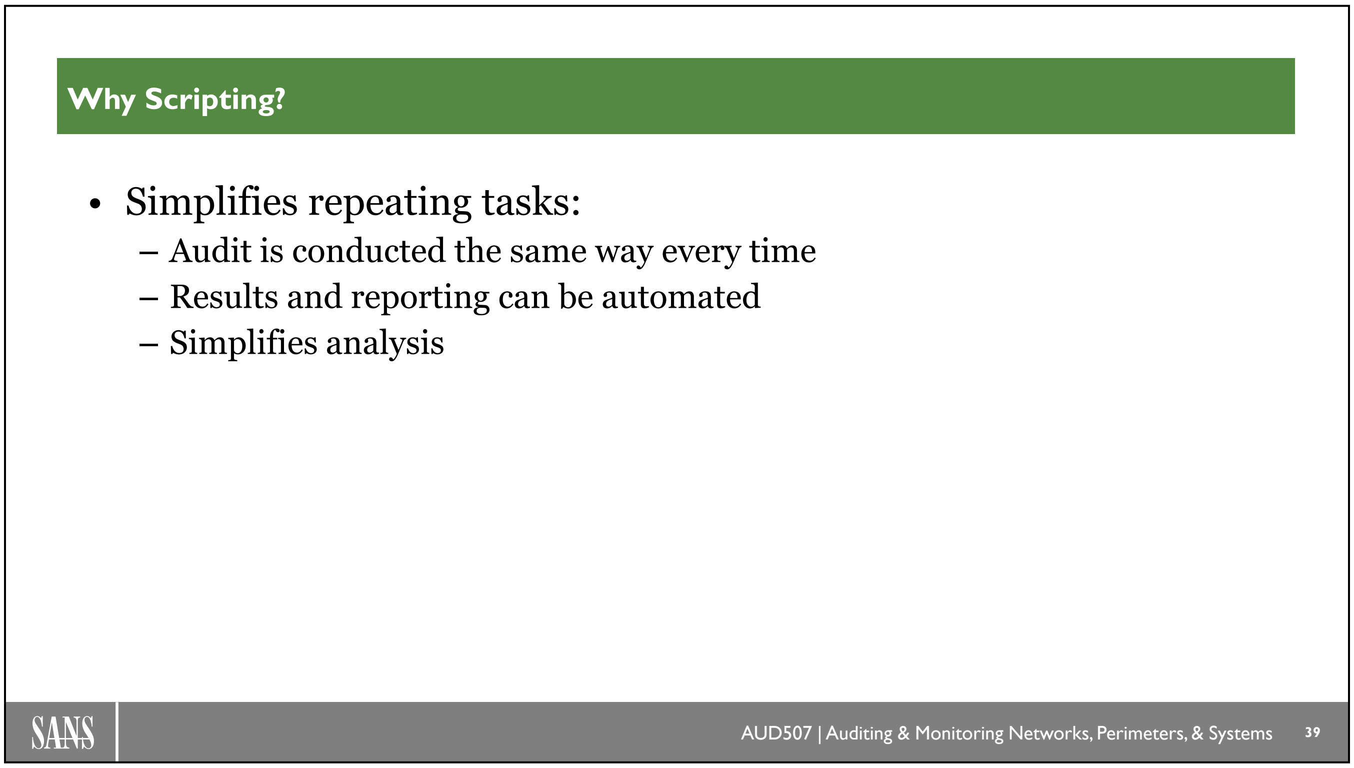
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Why Scripting?

- Simplifies repeating tasks:
  - Audit is conducted the same way every time
  - Results and reporting can be automated
  - Simplifies analysis

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 39

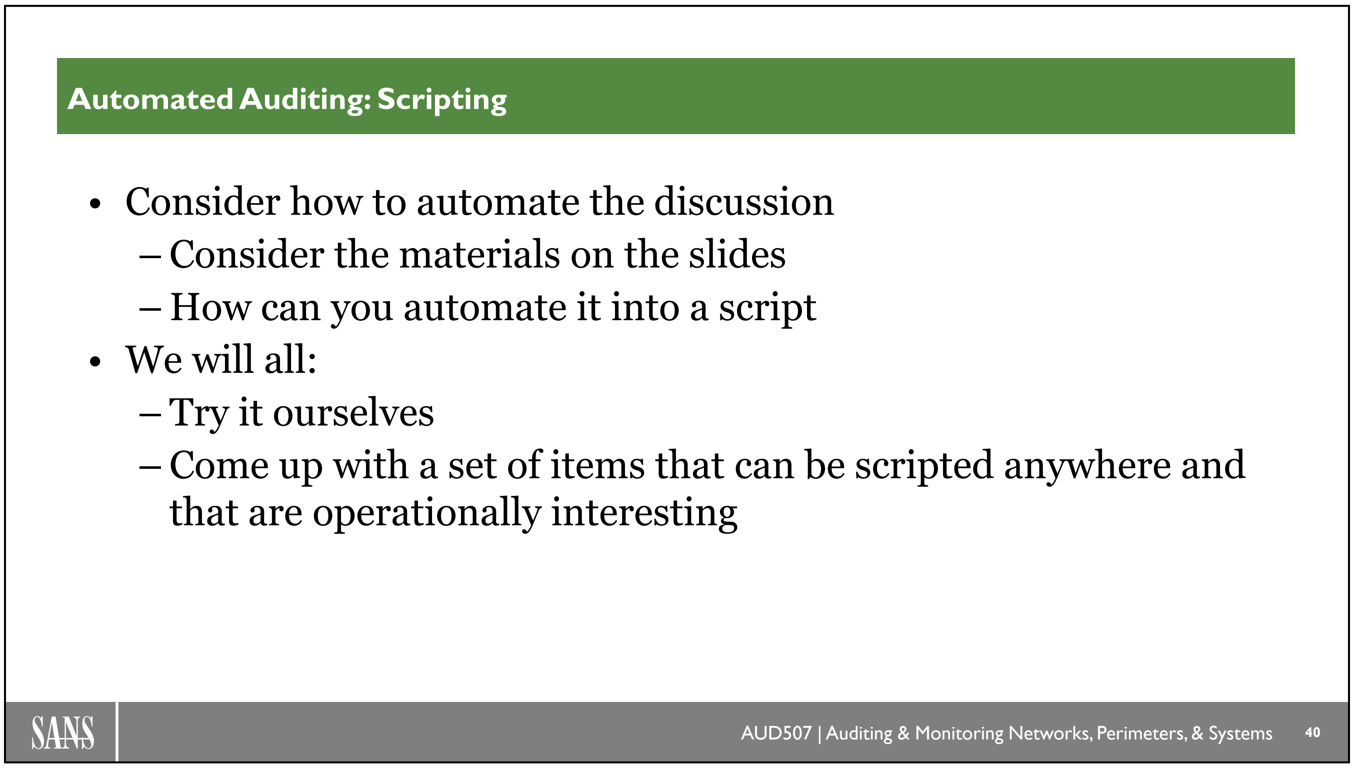
We've covered only a very few basics, but even so we have enough ammunition to write some fairly complete audit scripts. Remember that our purpose in covering this material is not to make you a system administrator or to turn you into a programmer. Our real goal is to allow us to create repeatable audits and automated audits.

Scripts can also be very useful when it comes to performing analyses. For instance, earlier in the week we looked at performing a firewall validation that would produce lots of output into a text file. Rather than trying to analyze all of that output by hand, we can write a very simple script to look for specific items or perhaps to summarize the results.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Automated Auditing: Scripting

- Consider how to automate the discussion
  - Consider the materials on the slides
  - How can you automate it into a script
- We will all:
  - Try it ourselves
  - Come up with a set of items that can be scripted anywhere and that are operationally interesting

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 40

In a few moments, you embark on a lab exercise to create a basic script. After that's done, we take a look at an example baseline audit script for UNIX machines that is installed on one of the course virtual machines.

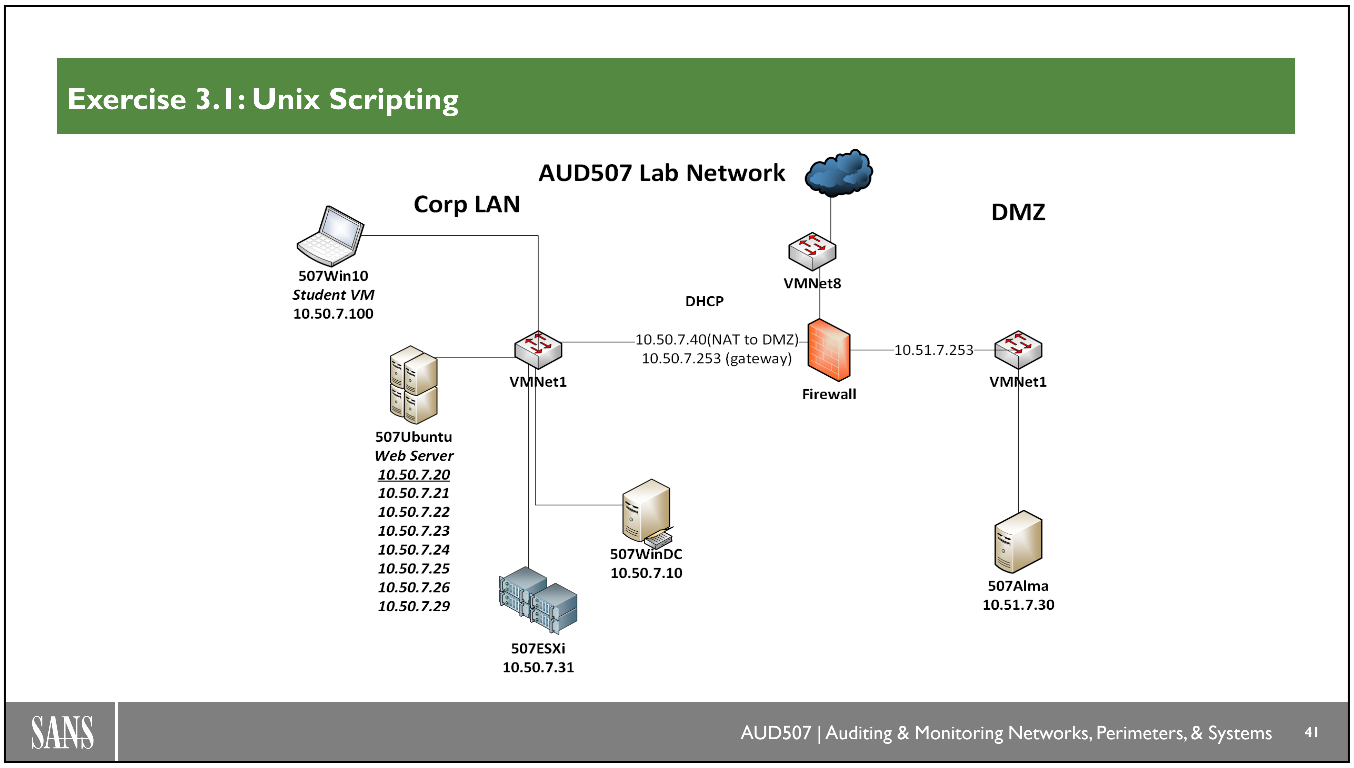
From there on out, we will encourage you to consider which of the various tests would be useful for you to include in a baseline system and to also think about how to automate those things. As an auditor, we are interested in this from the point of view of simplifying our process and creating consistency. From an operational standpoint, we're interested in passing this knowledge on or working with an administrator to create this system so that we have sustainable systems providing important information and feedback to administrators.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



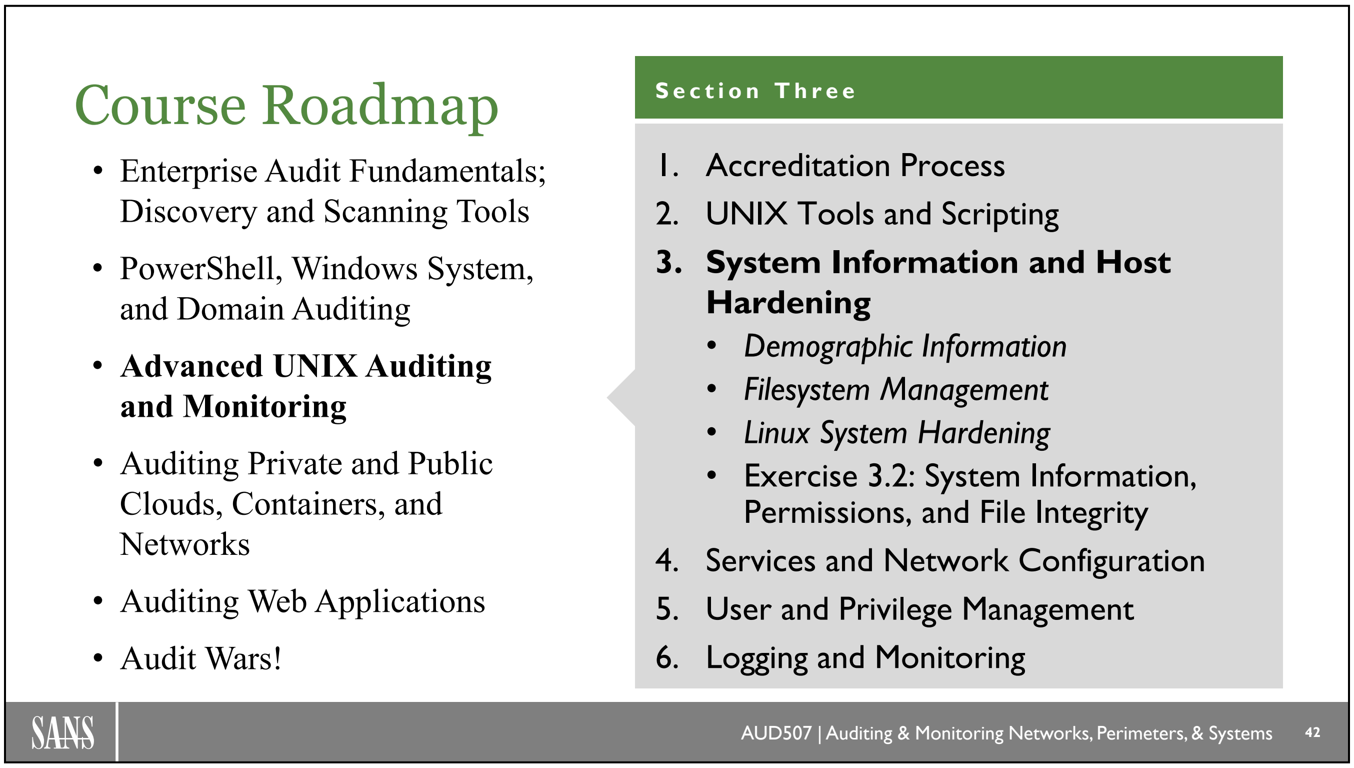
Exercise 3.1: Unix Scripting

Please get started with Exercise 3.1: Unix Scripting in the workbook.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Course Roadmap

- Enterprise Audit Fundamentals; Discovery and Scanning Tools
- PowerShell, Windows System, and Domain Auditing
- **Advanced UNIX Auditing and Monitoring**
- Auditing Private and Public Clouds, Containers, and Networks
- Auditing Web Applications
- Audit Wars!

Section Three

1. Accreditation Process
2. UNIX Tools and Scripting
3. **System Information and Host Hardening**
   - *Demographic Information*
   - *Filesystem Management*
   - *Linux System Hardening*
   - Exercise 3.2: System Information, Permissions, and File Integrity
4. Services and Network Configuration
5. User and Privilege Management
6. Logging and Monitoring

In this next section, we walk through a series of audit objectives and activities that will be used in the majority of UNIX audits. In the end, you will have a good framework of steps that can be taken to evaluate a UNIX system in your own environment.

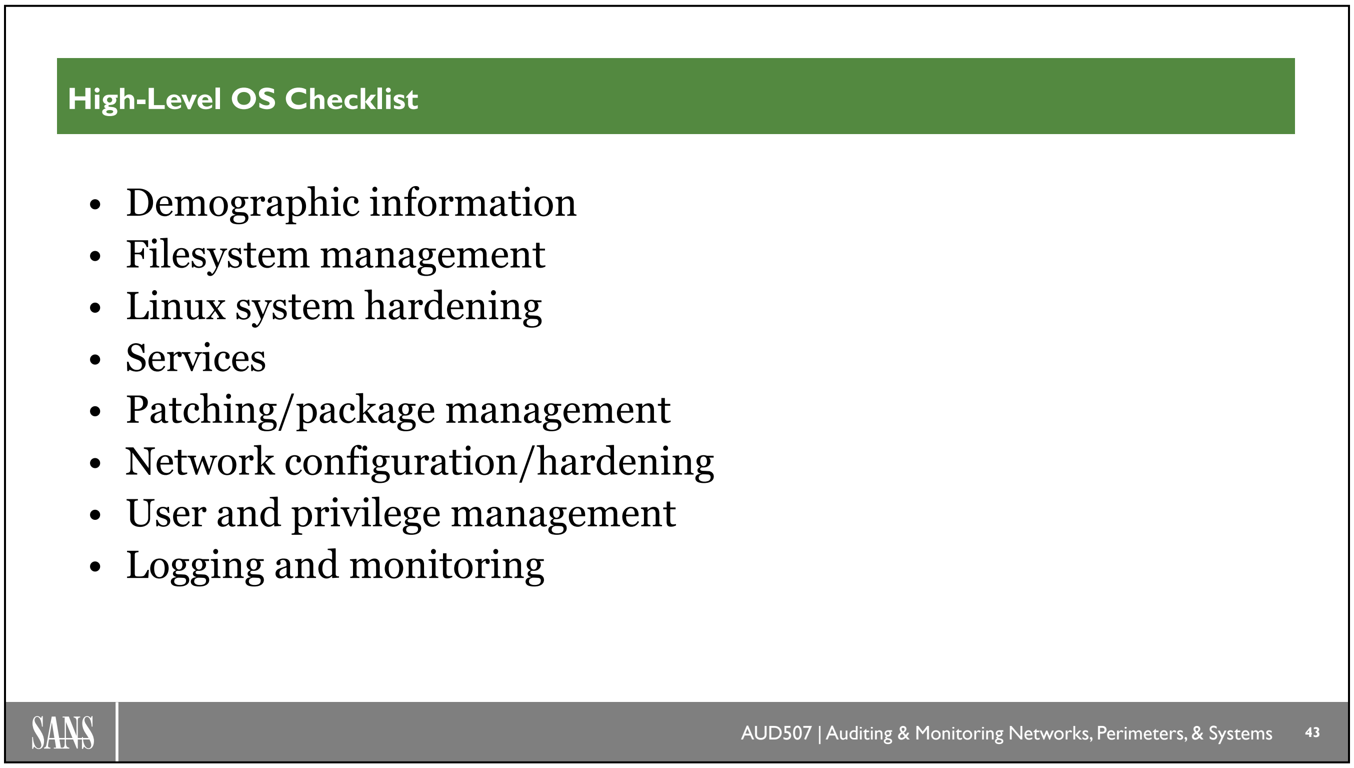
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



High-Level OS Checklist

- Demographic information
- Filesystem management
- Linux system hardening
- Services
- Patching/package management
- Network configuration/hardening
- User and privilege management
- Logging and monitoring

SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 43

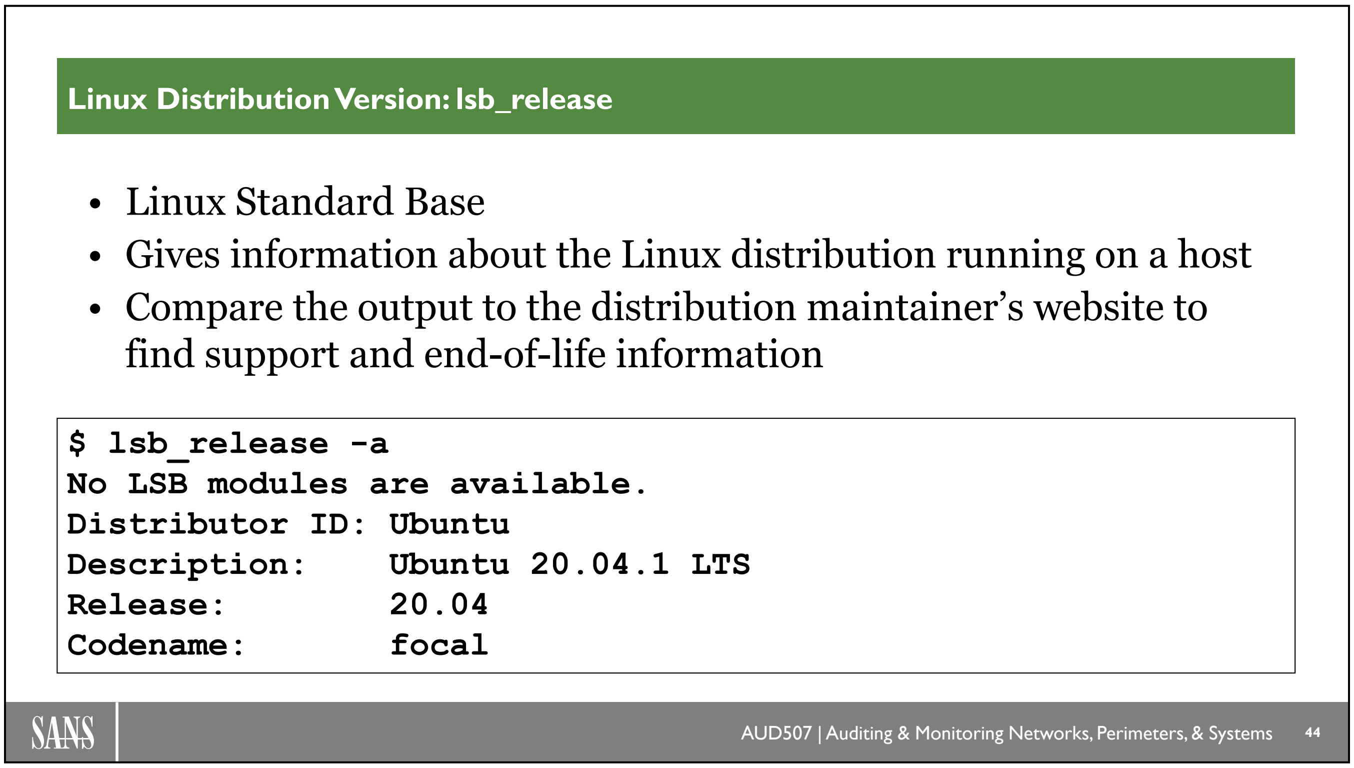
With all of the introductory material out of the way, we are now ready to conduct our research into auditing UNIX systems. To do this, we will be following the same outline that we did with Windows, which represents a high-level operating system checklist. Along the way, we will discuss how these items are expressed in UNIX systems, how to examine the relevant settings or configuration, and what sorts of things we should expect to see as good practice.

Remember that we have chosen to examine Windows and UNIX in some depth since these are the most common operating systems you will interact with. That said, this high-level checklist can be used with *any* operating system. You simply need to perform the research into the items in the checklist to determine how to extract the relevant information. Along the way, be sure to be on the lookout for operating system–specific issues.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Linux Distribution Version: lsb_release

- Linux Standard Base
- Gives information about the Linux distribution running on a host
- Compare the output to the distribution maintainer's website to find support and end-of-life information

```
$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.1 LTS
Release:        20.04
Codename:       focal
```

The lsb_release command prints information about the Linux distribution running on a host. This is a good way for an auditor to find the version number of the distribution so they can research its support and end-of-life status.

The options for running lsb_release are shown below:

```
lsb_release --help
Usage: lsb_release [options]

Options:
  -h, --help          show this help message and exit
  -v, --version       show LSB modules this system supports
  -i, --id            show distributor ID
  -d, --description   show description of this distribution
  -r, --release       show release number of this distribution
  -c, --codename      show code name of this distribution
  -a, --all           show all of the above information
  -s, --short         show requested information in short format
```

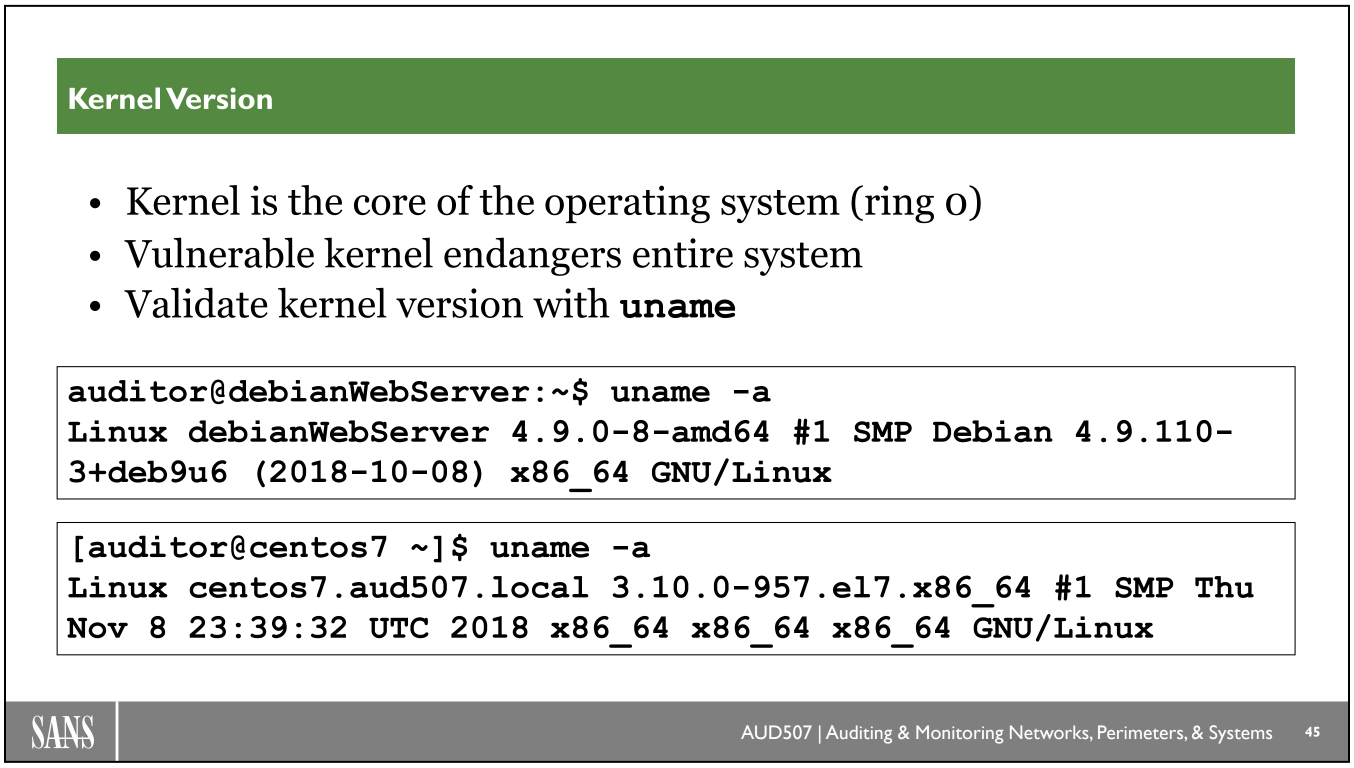
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Kernel Version

- Kernel is the core of the operating system (ring 0)
- Vulnerable kernel endangers entire system
- Validate kernel version with **`uname`**

```
auditor@debianWebServer:~$ uname -a
Linux debianWebServer 4.9.0-8-amd64 #1 SMP Debian 4.9.110-
3+deb9u6 (2018-10-08) x86_64 GNU/Linux
```

```
[auditor@centos7 ~]$ uname -a
Linux centos7.aud507.local 3.10.0-957.el17.x86_64 #1 SMP Thu
Nov 8 23:39:32 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 45

When an attacker compromises or attempts to compromise your system, one of the very first things that he or she will do is attempt to run the ‘uname’ tool. This tool reveals information about the kernel, the processor on the system, build time, and so on. Especially in a reconnaissance phase, this tool is invaluable for an attacker if he can run it because it allows him to narrow the field to exploits that are likely to succeed against the architecture in question.

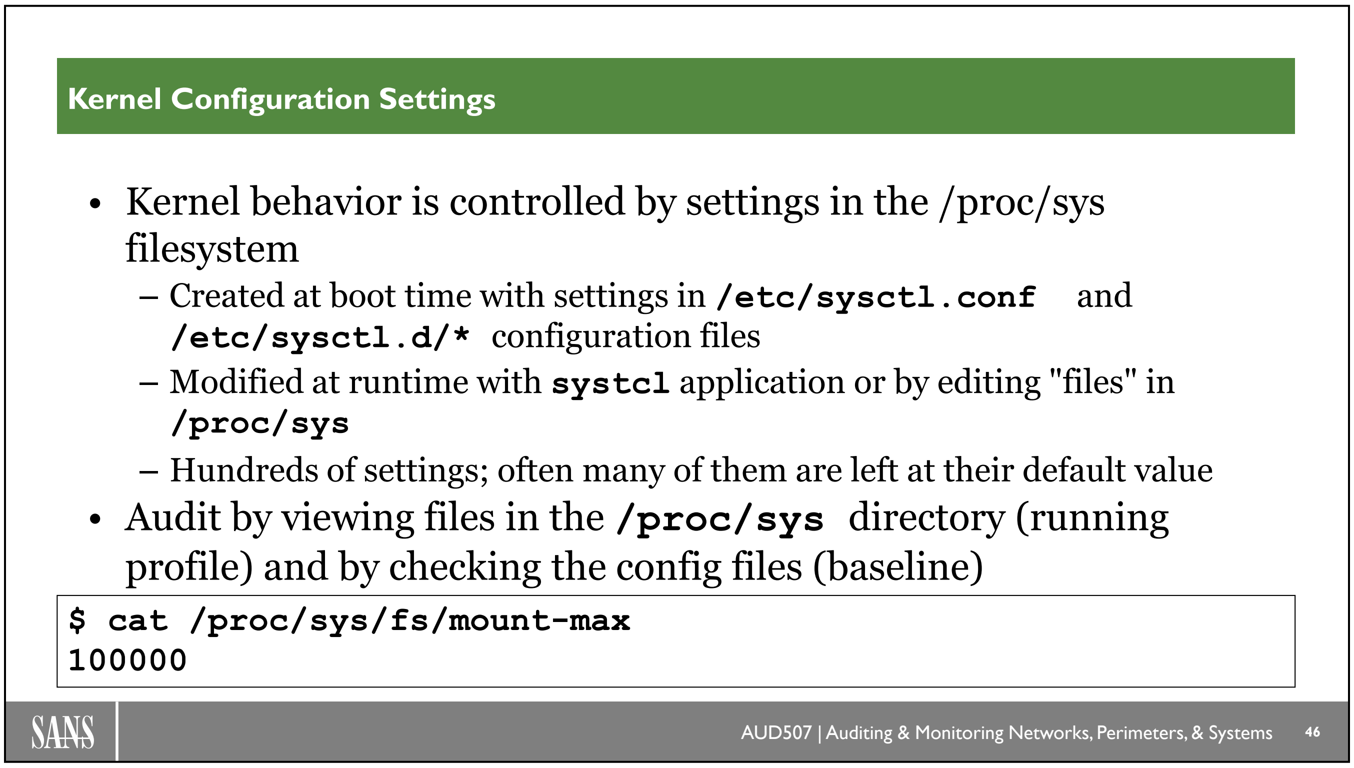
For an auditor, the tool is useful because it allows the auditor to gather baseline information about the machine. Note that we can see the type of UNIX kernel in use, the precise version, and the date on which the kernel was compiled. It is easy to assume that an old kernel version number or compilation date indicates that the system is out of date or has not been patched. While this is possible, it would be important to verify whether or not the particular kernel version in use does have vulnerabilities before we report that it is a risk because it is old.

Note the disparity in version numbers across these two hosts, even though the kernels were compiled within a month of each other. Some Linux distributions use older, stable code bases, while other tend to use more cutting-edge code. As long as the kernel in use is current, supported, and has no known vulnerabilities, the system should be considered properly patched.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Kernel Configuration Settings

- Kernel behavior is controlled by settings in the /proc/sys filesystem
  - Created at boot time with settings in **`/etc/sysctl.conf`** and **`/etc/sysctl.d/*`** configuration files
  - Modified at runtime with **`systcl`** application or by editing "files" in **`/proc/sys`**
  - Hundreds of settings; often many of them are left at their default value
- Audit by viewing files in the **`/proc/sys`** directory (running profile) and by checking the config files (baseline)

```
$ cat /proc/sys/fs/mount-max
100000
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 46

Since the kernel is an integral piece of the operating system, understanding how it is configured will be very important to us as we audit the system. The settings for the kernel are managed by sysctl (system control), and are loaded from the /etc/sysctl.conf file and the associated files stored under the /etc/sysctl.d/ directory. While the system is running system control settings may be read and/or written using the /proc/sys pseudo-filesystem.

To audit the settings, you will check the configuration files to see what the baseline settings are for the system (any settings not defined in the configuration files will be populated in a default state) and you will read the contents of the /proc/sys files. In the screenshot we are checking to see the maximum number of filesystems which can be mounted on this host.

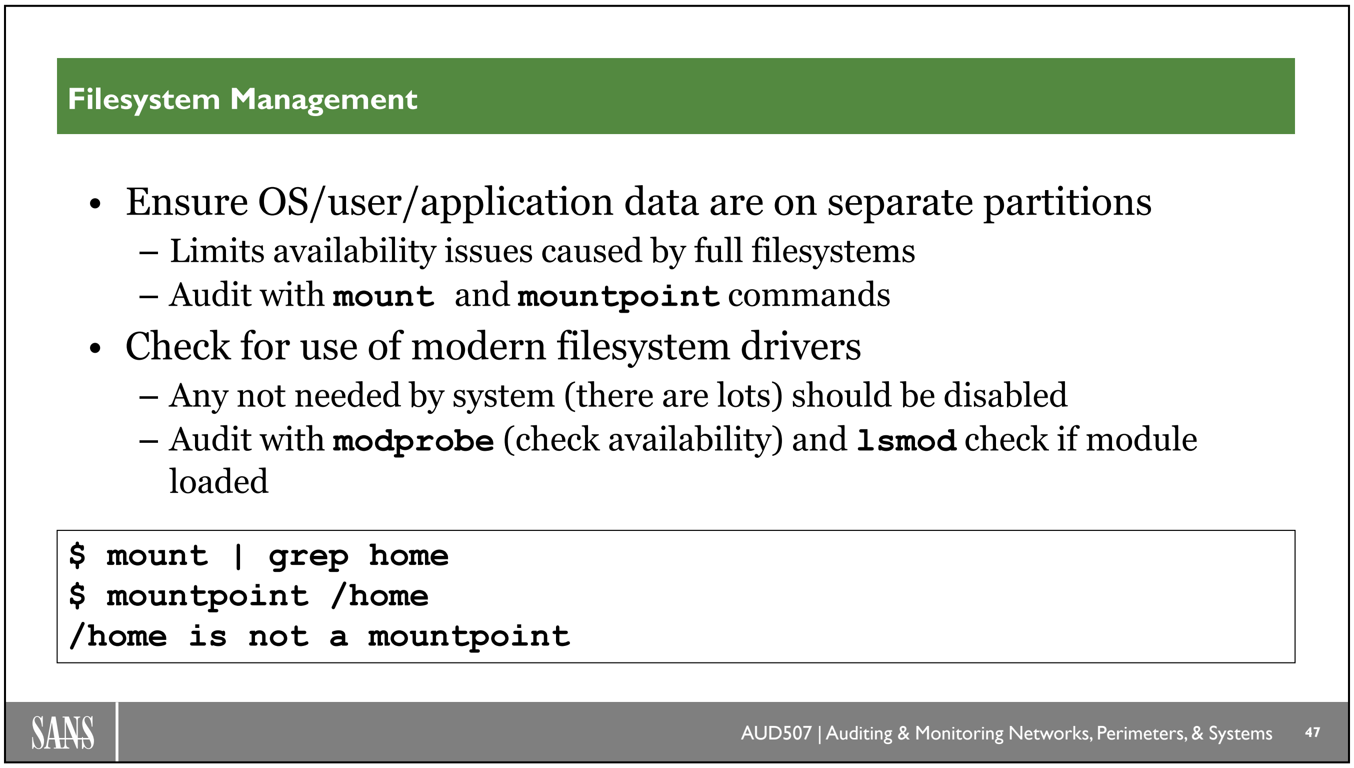
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Filesystem Management

- Ensure OS/user/application data are on separate partitions
  - Limits availability issues caused by full filesystems
  - Audit with **`mount`** and **`mountpoint`** commands
- Check for use of modern filesystem drivers
  - Any not needed by system (there are lots) should be disabled
  - Audit with **`modprobe`** (check availability) and **`lsmod`** check if module loaded

```
$ mount | grep home
$ mountpoint /home
/home is not a mountpoint
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 47

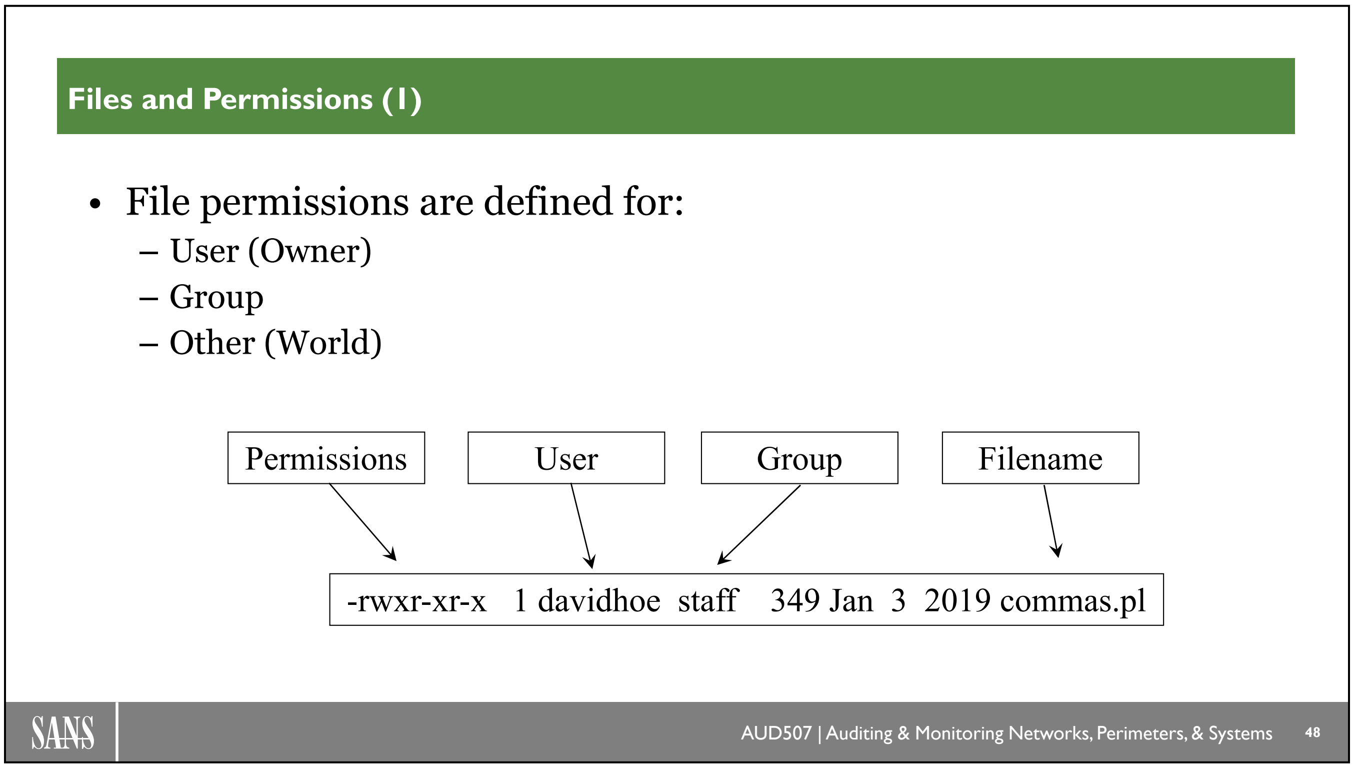
Many of the directories in the *nix filesystem are dedicated for use by users or application programs. One user or application should not be able to take the whole system down by filling the entire disk. To protect against this, our systems should have directories which are susceptible to filling up mounted to dedicated disk partitions. This will limit the "blast radius" of a full directory and will protect the larger system. You can audit these directories using the **`mount`** command to list which directories are mounted to disk partitions and the **`mountpoint`** command to test whether a specific directory is a mountpoint.

On the topic of filesystems, Linux has support available for a large number of disk formatting schemes. As a matter of economy of mechanism, your administrators should enable only the filesystem types which are required for organizational purposes. Test for this by first checking whether the kernel modules which support a filesystem are available for loading into the kernel. The **`modprobe`** "dry run" option is perfect for this: **`modprobe -n -v driverName`**. If the module *is* available, then test to ensure that is not loaded into the running kernel using the **`lsmod`** command, e.g., **`lsmod | grep udf`** for the UDF filesystem driver.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Files and Permissions (1)

- File permissions are defined for:
  - User (Owner)
  - Group
  - Other (World)

Because we're already on the topic of files, it seems that we should mention something about access controls or file permissions. Although many modern operating systems have very complex and granular mechanisms for granting or restricting access to files and other resources, UNIX uses a very simple system. In terms of permissions, it is only possible to specify access controls in terms of what the user (owner) can do, what members of the group that owns the file can do, or what the world (or "others") can do.

In its simplest form, file permissions amount to Read, Write, and Execute permissions. There can be some interesting side effects to these permissions that we'll get to shortly, but in essence we are able to define the access the owner, group, or world have to a file based on those simple terms. There are a few other special options that can be used with file permissions as well.

It is possible to set the permissions on a file so that the person running the system inherits the effective user ID (EUID) of the user who owns the file. This is known as a set-uid (SUID) program. Essentially, this means that the process thread under which the program runs has all of the rights and permissions of the owner of the file. The /bin/passwd program, for example, is typically an SUID program owned by root; this is more or less required because only the root user may alter the password file on the system.

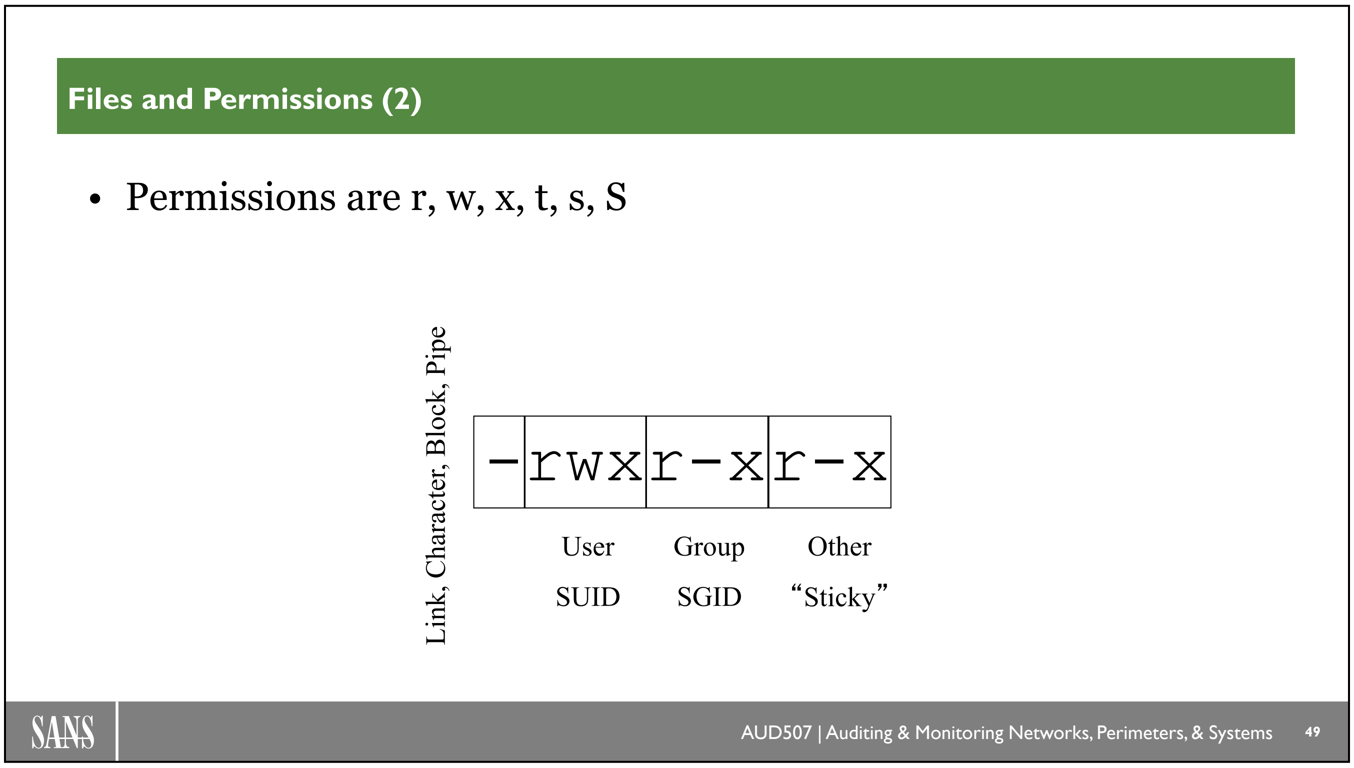
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

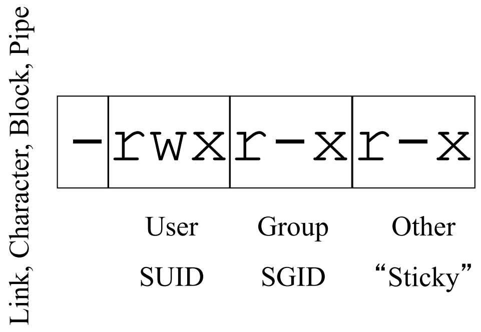
Technet24

© SANS Institute 2021



Files and Permissions (2)

- Permissions are r, w, x, t, s, S



Because there is only so much room for representing the file permissions, some of the positions do double duty. Logically, because an SUID program changes the EID, the 'x' for execute in the Owner section is replaced with an 's' for SUID. Similarly, if this were SGID, the Group 'x' would be replaced with an 'S'. The world 'x' is used to represent whether or not the 'sticky' bit is set. The sticky bit is mostly historical, though there are some modern uses. Unfortunately, interpretation of the sticky bit is operating system specific, so we will not attempt to describe all of the possible functions of this bit.

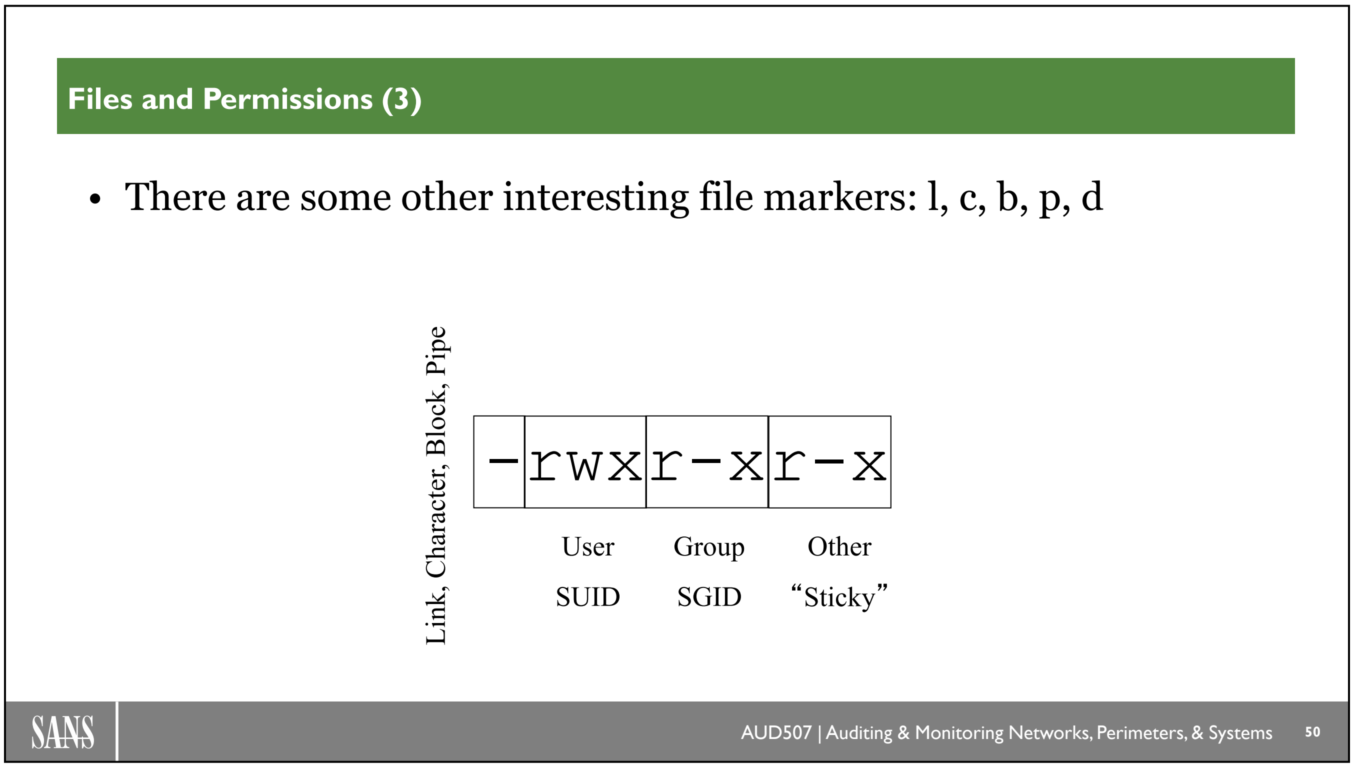
The most common usage of this bit today is for directories. Imagine that you have a server, perhaps a web server, where multiple users are involved in editing and uploading files to the shared web directory. An extremely common problem is that one of the users replaces a file in that directory and the default UMASK (user permissions mask) that is applied sets the permissions in such a way that none of the other authorized individuals can modify the file. At the same time, the file is set to be owned by the default group that the user is a member of, so the group permissions are also too restrictive.

To resolve this problem, one of the users reaches out to the administrator to have the group ownership of the file changed. Although this is a short-term fix, it is really not the correct solution. The best solution is to set the sticky bit on the *directory* that the file resides in. The sticky bit used in this way will enforce the group ownership of all of the files in this directory. What this means is that when a new file is created by a user, that file will have the folder's group ownership regardless of what the default user's group is!

August 10, 2021

August 10, 2021

© SANS Institute 2021



Files and Permissions (3)

- There are some other interesting file markers: l, c, b, p, d

Link, Character, Block, Pipe
-rwxr-xr-x
User
Group
Other
SUID
SGID
“Sticky”
SANS
AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems
50

The first character listed with the file permissions is not actually used to define the permissions but the actual type of file. The five possible values for this position are l, c, b, p, and d. Each of these values is used to indicate that this is a “special” file.

Intuitively, the letter ‘d’ in this position indicates that this file is a directory. The other values are equally intuitive in that they represent the first letter of the term they replace:

l = Link

c = Character device

b = Block device

p = Named pipe

The only one of these types that is of special interest to us is a link or symbolic link. Links are used to create pointers to files within the filesystem or even to files on other physical filesystems. There are actually two types of links: Hard links and soft links. A discussion of the actual differences between these two is beyond the scope of our discussion, but the purpose of links is to allow you to create an easy reference point to a file that is physically located elsewhere. Effectively, this allows the file to appear to exist in more than one place at a time without consuming additional resources.

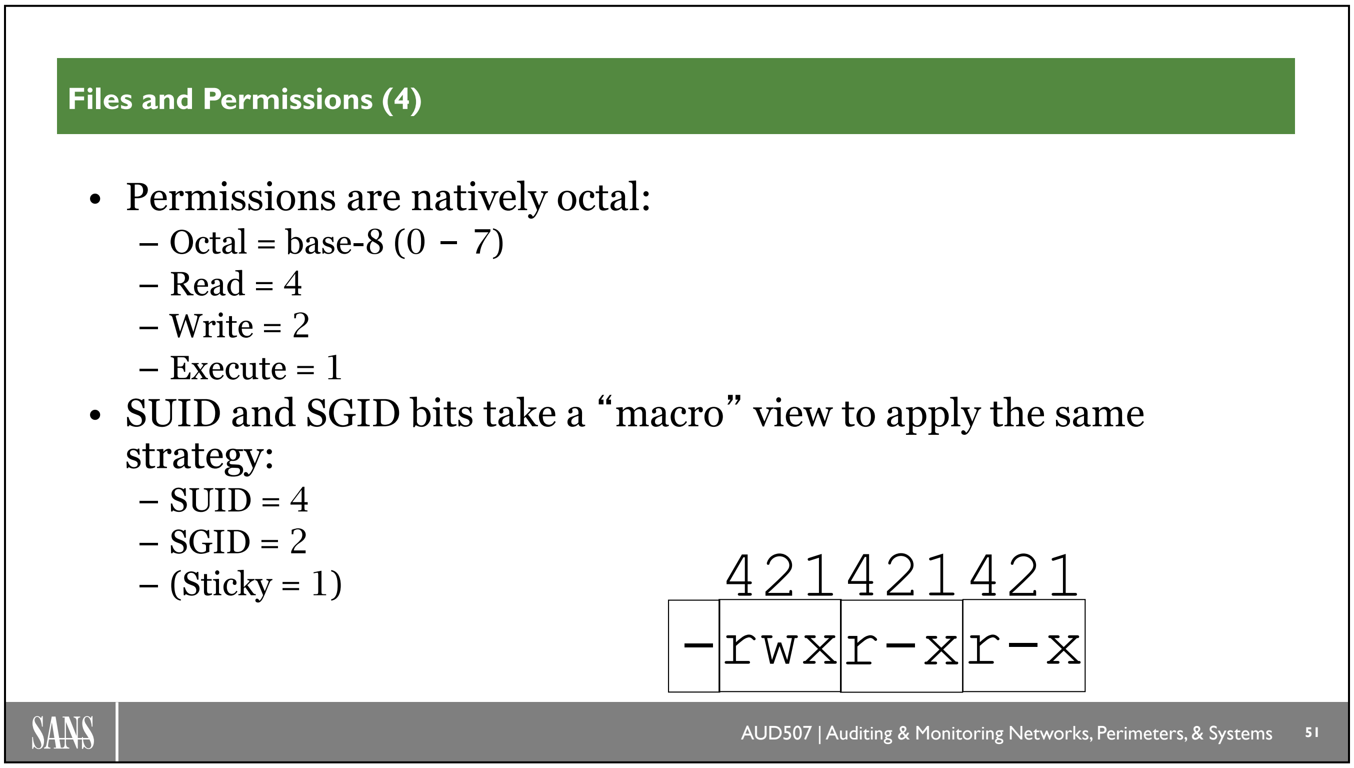
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Files and Permissions (4)

- Permissions are natively octal:
  - Octal = base-8 (0 – 7)
  - Read = 4
  - Write = 2
  - Execute = 1
- SUID and SGID bits take a “macro” view to apply the same strategy:
  - SUID = 4
  - SGID = 2
  - (Sticky = 1)

```
 421421421
-rwxr-xr-x
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 51

Within the filesystem itself, the file permissions are not actually stored as letters, but as base-8 or octal digits. This allows us to use any of three possible bit values to represent each of the read, write, or execute permissions. This means that if a file has both read and write permissions turned on but execute permission turned off, the value for that permission would be 4+2 = 6 (Read = 4, Write = 2). This also means that we can have an octal value from 0 through 7 in any of the three positions that define the permissions for the owner, the group, or the world.

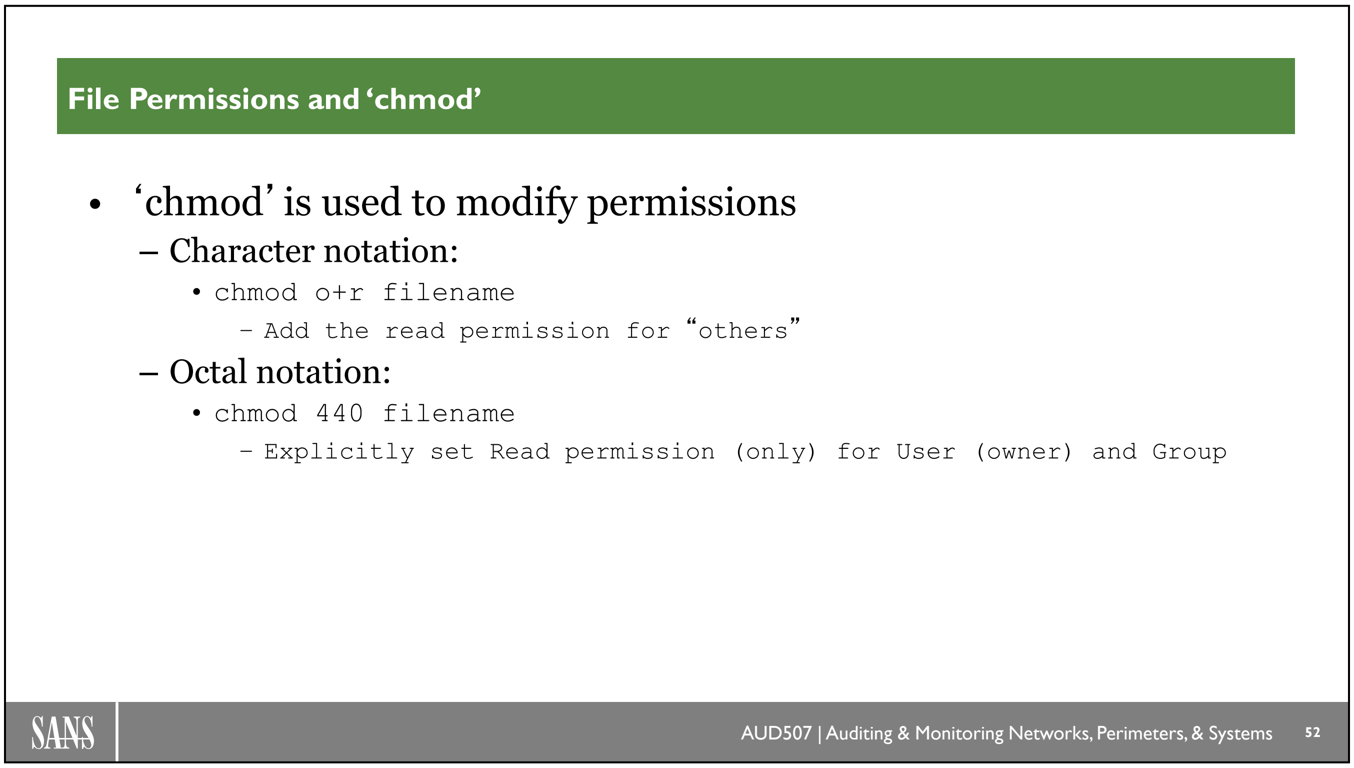
In reality, there is a fourth octal value available at the most significant or leftmost position. This octal value is used to control the special SUID, SGID, and sticky bits. In a somewhat abstract yet logical way, the value of the SUID, which makes us appear to be the owner, is 4, the SGID bit is valued at 2, and the sticky bit is valued at 1. In other words, if we look at the overall permissions structure as three possible bit values (owner, group, world), we simply assign the proper bit positional value and arrive at the method used internally to represent these bits.

I know that sounds really technical, but I mention it now because much later today, you will need to understand how these bits work to use an extremely important tool: find.

August 10, 2021

August 10, 2021

© SANS Institute 2021



File Permissions and 'chmod'

- 'chmod' is used to modify permissions
  - Character notation:
    - `chmod o+r filename`
      - Add the read permission for "others"
  - Octal notation:
    - `chmod 440 filename`
      - Explicitly set Read permission (only) for User (owner) and Group

The command-line mechanism for adjusting the permissions that apply to a file or directory is the 'chmod' utility. When using this utility, we have the ability to specify the permissions to set or change on the file in terms of a simple character notation or by using an octal representation (remember that octal stuff from the last slide? ☺).

Using the character representation, it is a simple matter to add or remove specific permissions. The octal representation requires that we specify exactly what we would like the permissions to be. Here's the general format used for setting symbolic file permissions:

chmod [u|g|o|a[u|g|o[u|g|o]]][+|-|=][r|s|t|w|x|X|u|g|o] [filespec]

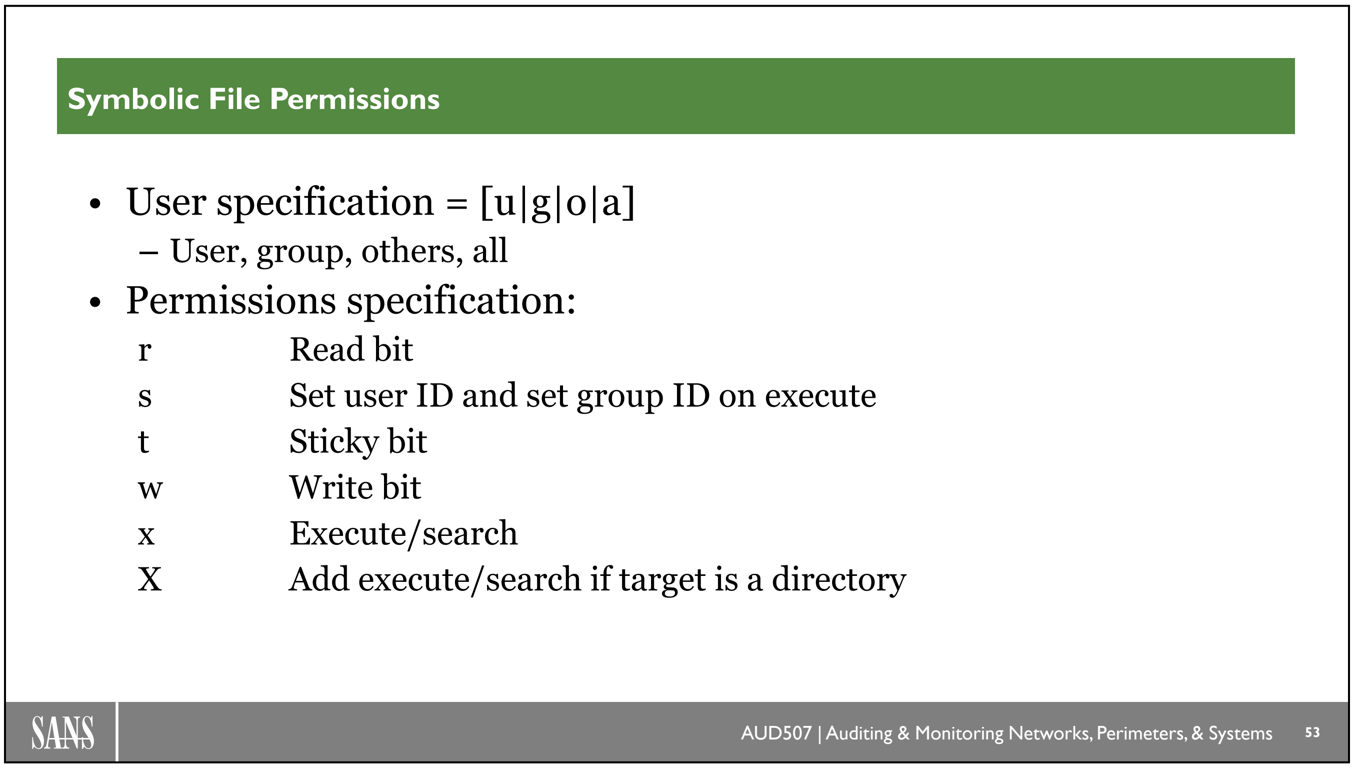
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Symbolic File Permissions

- User specification = [u|g|o|a]
  - User, group, others, all
- Permissions specification:

r	Read bit
s	Set user ID and set group ID on execute
t	Sticky bit
w	Write bit
x	Execute/search
X	Add execute/search if target is a directory

SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 53

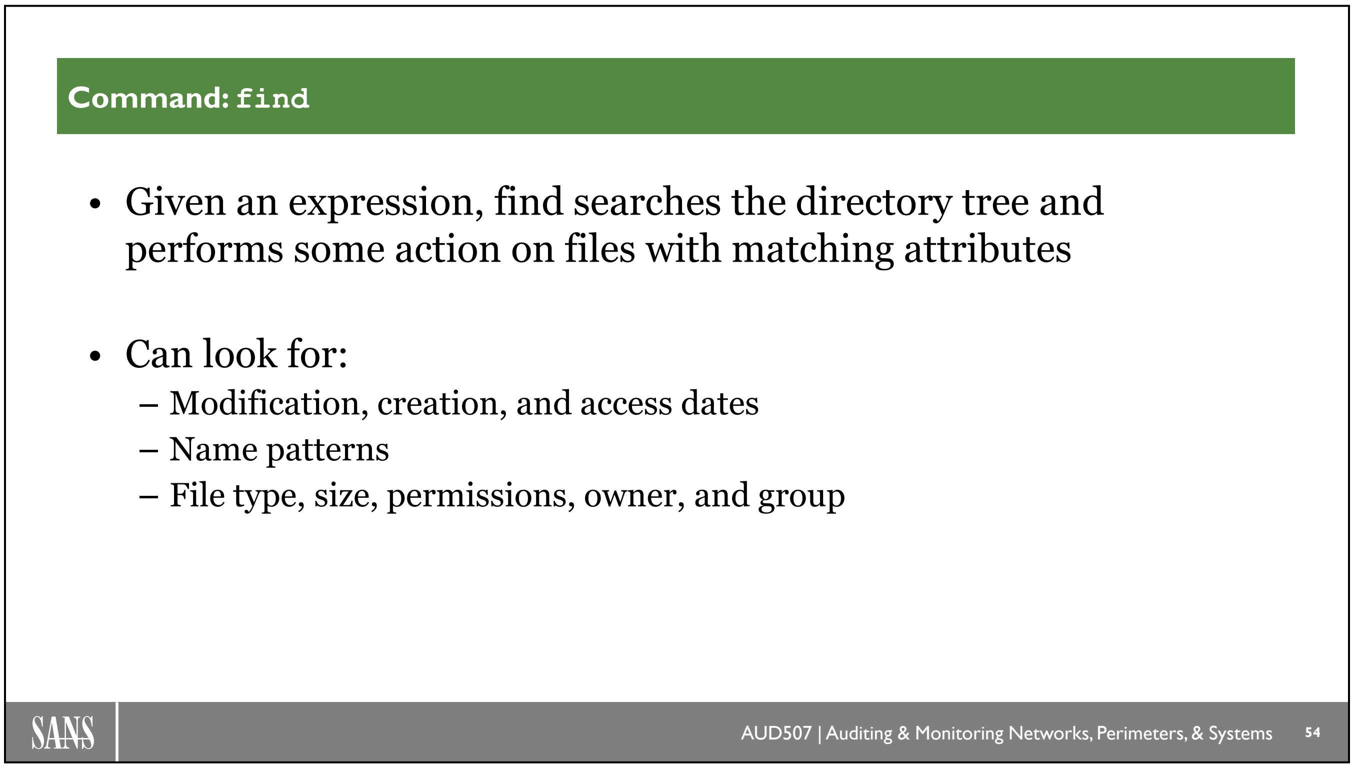
The symbolic permissions are more flexible than the numeric permissions in that it is easier to modify a single bit or selective set of bits across a selection of files without changing the permissions to an explicit set of permissions. When making changes, we can specify those changes to apply to the owner (u), the group (g), others (o), or any combination of these. To change all three, we can use the 'a' symbol to represent that we're changing all of them. 'a' is a synonym for 'ugo'.

The actual permissions are pretty easy to understand. We can choose to add (+), remove (-), or set explicitly (=) any of the typical read (r), write (w), and execute (x) permissions. There are also symbols to allow us to set the sticky bit, which is appropriately represented by a 't' because that is how it is represented in the file permissions display from 'ls'. We can also turn on the set user and set group ID bits (s), though there are no consistent settings that allow us to target just the set group or just the set user ID bit. Finally, there's a pretty handy 'X' permission that will set the execute permission only if the target is a directory.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Command: `find`

- Given an expression, find searches the directory tree and performs some action on files with matching attributes
- Can look for:
  - Modification, creation, and access dates
  - Name patterns
  - File type, size, permissions, owner, and group

Find is an incredible command. The degree to which one can define filenames and file classes is astounding. And better yet, for each matching file that is found, an action can be performed. This is incredibly useful as a system maintenance feature, but it's also incredibly dangerous. Be careful performing some action on files returned.

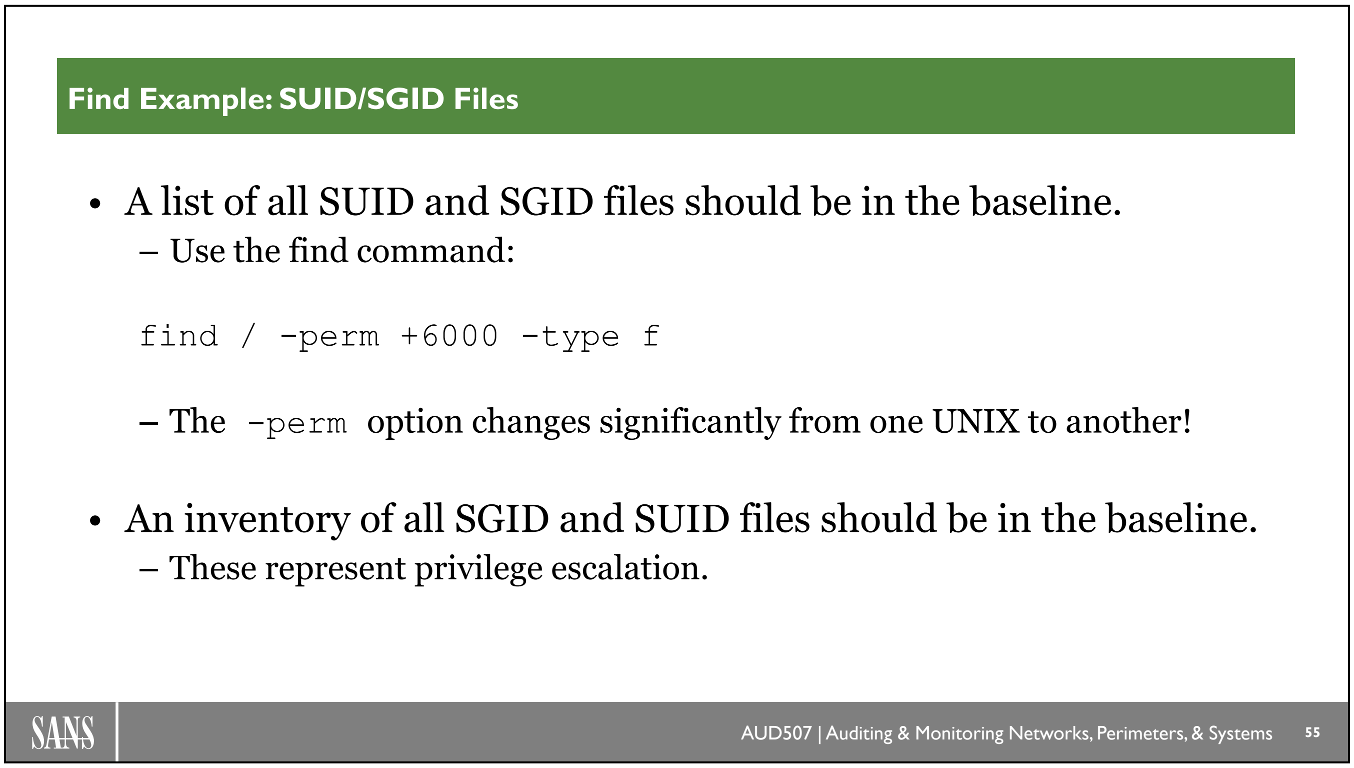
The real power, however, is that we can search for files based on pretty much any file-related criteria you can imagine. This includes the kind, size, creation time, modification time, access time, permissions, owner, group, relative modification or access times in days or minutes, type of filesystem that the file resides on, filename patterns, inode number, file path, or group.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Find Example: SUID/SGID Files

- A list of all SUID and SGID files should be in the baseline.
  - Use the find command:

```
find / -perm +6000 -type f
```

  - The `-perm` option changes significantly from one UNIX to another!
- An inventory of all SGID and SUID files should be in the baseline.
  - These represent privilege escalation.

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 55

To audit SUID root files, use the find command with a few extra parameters. This incantation tells find to start at the root directory, /, and recursively search the filesystem looking for any files that are SUID or SGID and of type file. How did we arrive at SUID or SGID?

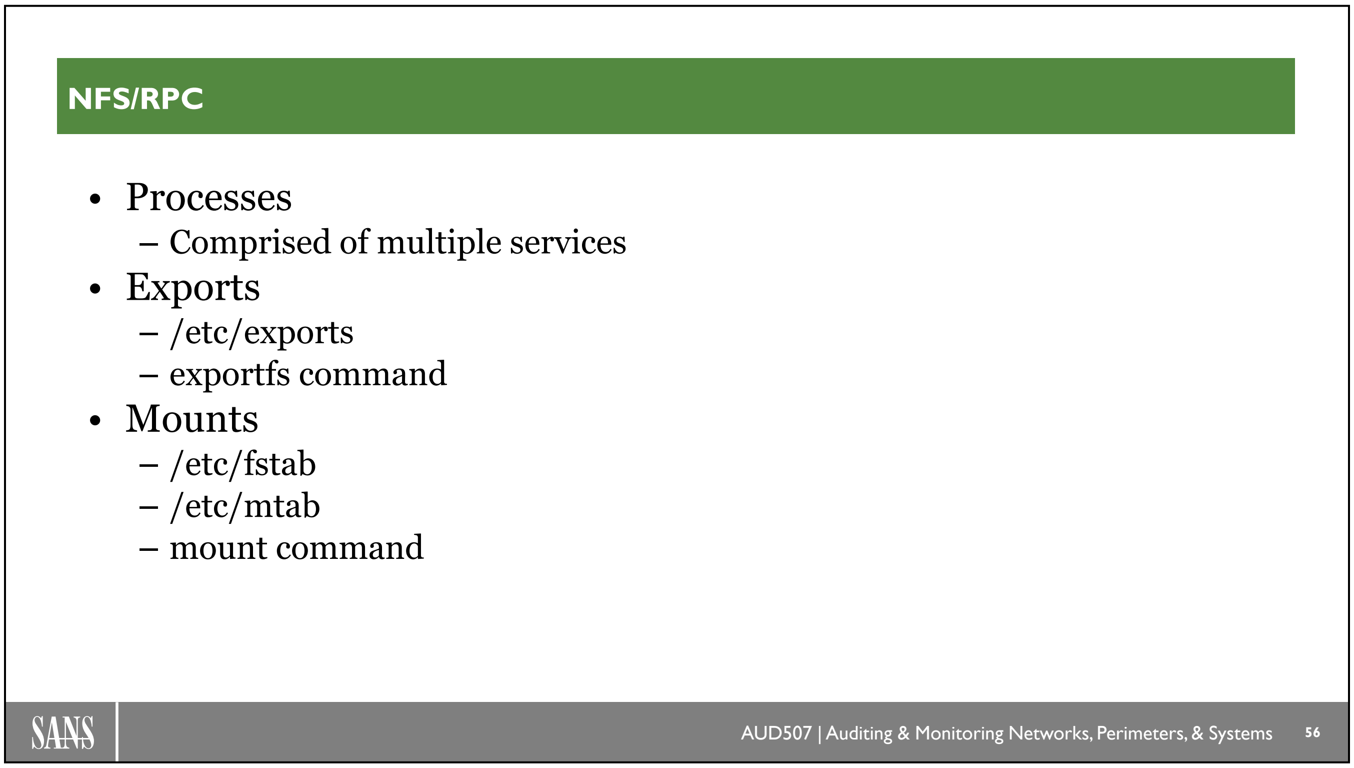
The typical file permissions are represented by three octal digits (for example, 755). These permissions apply to the user who owns the file, the group that owns the file, and others (or the world). A sort of "super-permission" sits above these. The 1 bit represents the sticky bit, the 2 bit represents SGID, and the 4 bit represents SUID. By searching for "+6000" we are indicating that we want files where either SUID, SGID, or both are set.

One of the first things incident handlers will do when I suspect that a system has been compromised is to check the current list of SUID files against the master that I created and stored on a CD, USB stick, or read-only network share. Why do they store the SUID file listing on some other device or system? This is a special measure that should be taken to ensure that the contents of this file cannot be altered by a remote hacker.

August 10, 2021

August 10, 2021

© SANS Institute 2021



NFS/RPC

- Processes
  - Comprised of multiple services
- Exports
  - /etc/exports
  - exportfs command
- Mounts
  - /etc/fstab
  - /etc/mtab
  - mount command

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 56

A top issue for both Windows and UNIX concerns weak or nonexistent file-share permissions. Although it is absolutely true that a UNIX system could be using SAMBA to create Windows-style shares, in our experience this isn't where the problems are. If an administrator is using SAMBA, he is usually using it to connect his UNIX system to a domain share to access files, not to share folders to the Windows network. This isn't to say that it can't happen, though, so if we found SAMBA running, we should certainly inquire and check on the file share permissions.

In our experience, the weak file-share permission issue in UNIX typically involves NFS. We see UNIX systems used as a sort of network "duct tape" to connect two disparate systems, allowing for data to be transferred. For example, in one enterprise, we helped it to create a transport between its brand-new ERP system and its legacy shipping system. What was the transport? A UNIX system.

"Pick tickets" were generated on the ERP system and dropped into an NFS share. The UNIX system would pick them up, process them, and then share them with the shipping system. Why NFS? Because it's the native file-sharing application for UNIX. It is a Remote Procedure Call (RPC) service made up of a number of services like nfsd, mountd, statd, and others. It typically has a configuration file called "/etc/exports" and will often create entries in the /etc/fstab for frequently mounted NFS volumes.

Although we're dealing with an RPC (NFS is an RPC service), let's talk about RPCs in general. A Remote Procedure Call is a well-defined system for registering available services that are essentially published, or at least available, on the network. We can query a system to determine which RPC services are available by running the command 'rpcinfo –p'. If the system has any RPCs installed, this tool will almost always be installed. We can also install this tool on a system and then query remote systems. When running this tool, the results will reveal any RPC programs (RPCs are identified by unique program numbers, not by port numbers) that are available on the system. These should be minimized and should absolutely be a part of any good baseline of the system.

August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021

Checking for NFS Server

- Check the contents of the /etc/exports configuration file
- Run the exportfs command

```
# cat /etc/exports
/mnt/Z3-32TB/ESXiShare  -alldirs -mapall="nobody":"nogroup"
-network 172.18.0.0/16

/mnt/Z3-32TB/XenShare  -alldirs -mapall="nobody":"nogroup"
-network 172.18.0.0/16
```

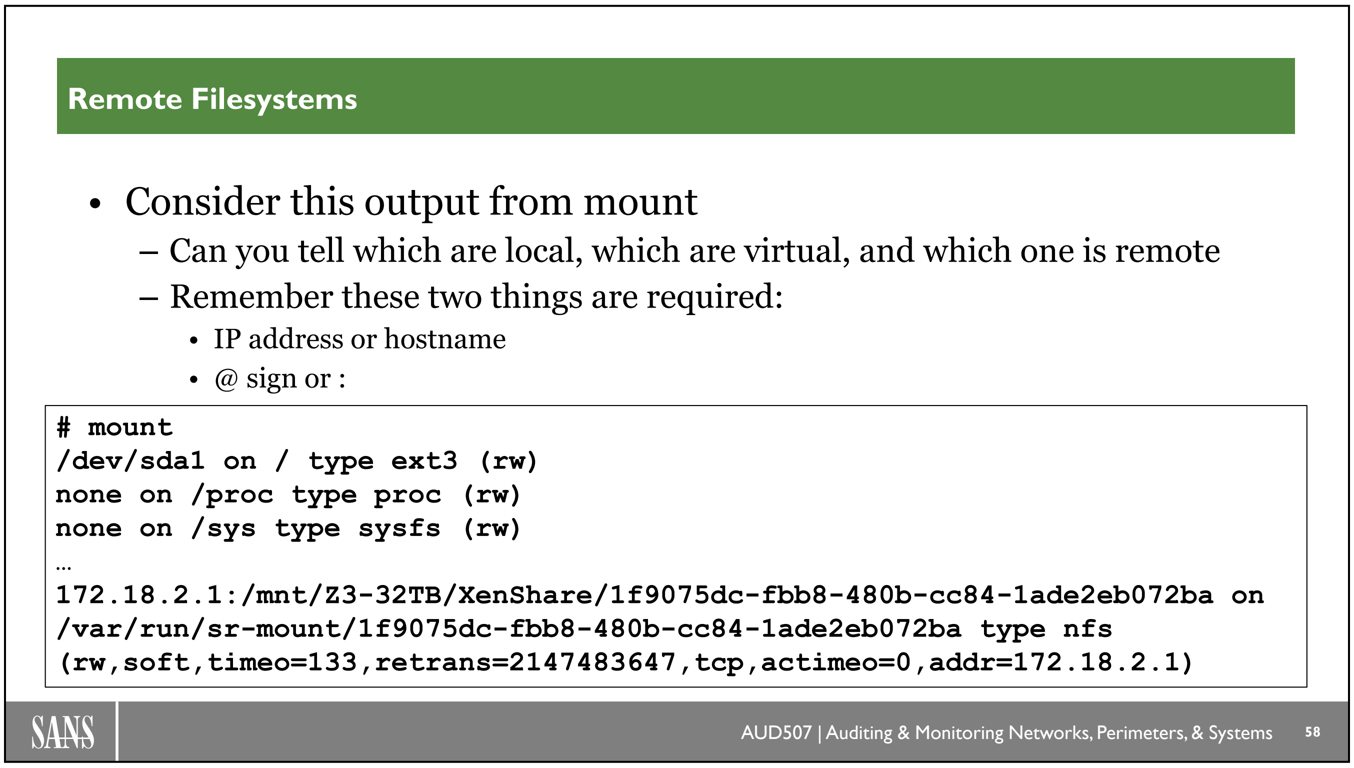
Now that we're aware of what NFS is and how it works at a high level, we can look for signs of it. In this case, we checked for the /etc/exports file, and saw that it contains information about two shares. This machine is definitely configured to export two different filesystems.

We could also run the "exportfs" command on a Linux host to see if any shares are currently being exported.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Remote Filesystems

- Consider this output from mount
  - Can you tell which are local, which are virtual, and which one is remote
  - Remember these two things are required:
    - IP address or hostname
    - @ sign or :

```
# mount
/dev/sda1 on / type ext3 (rw)
none on /proc type proc (rw)
none on /sys type sysfs (rw)
…
172.18.2.1:/mnt/Z3-32TB/XenShare/1f9075dc-fbb8-480b-cc84-1ade2eb072ba on
/var/run/sr-mount/1f9075dc-fbb8-480b-cc84-1ade2eb072ba type nfs
(rw,soft,timeo=133,retrans=2147483647,tcp,actimeo=0,addr=172.18.2.1)
```

SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 58

When looking at the fstab, the mtab, or the output from mount, you are looking for filesystems whose device has an @ or a : in it with a hostname or IP address. Look at the output of mount in the slide. Which of those filesystems are local? Which ones are virtual filesystems? Which of these is a remote filesystem?

Notice that one of the lines begins with "/dev". That's a dead giveaway that this is a physical drive. Two of them begin with the word "none". These are pretty clearly not devices. These are virtual filesystems of some kind.

Notice the last entry, though. This one has an IP address, followed by a colon and a file path: "172.18.2.1:/mnt/Z3-32TB/XenShare/1f9075dc-fbb8-480b-cc84-1ade2eb072ba." This is a mounted NFS share.

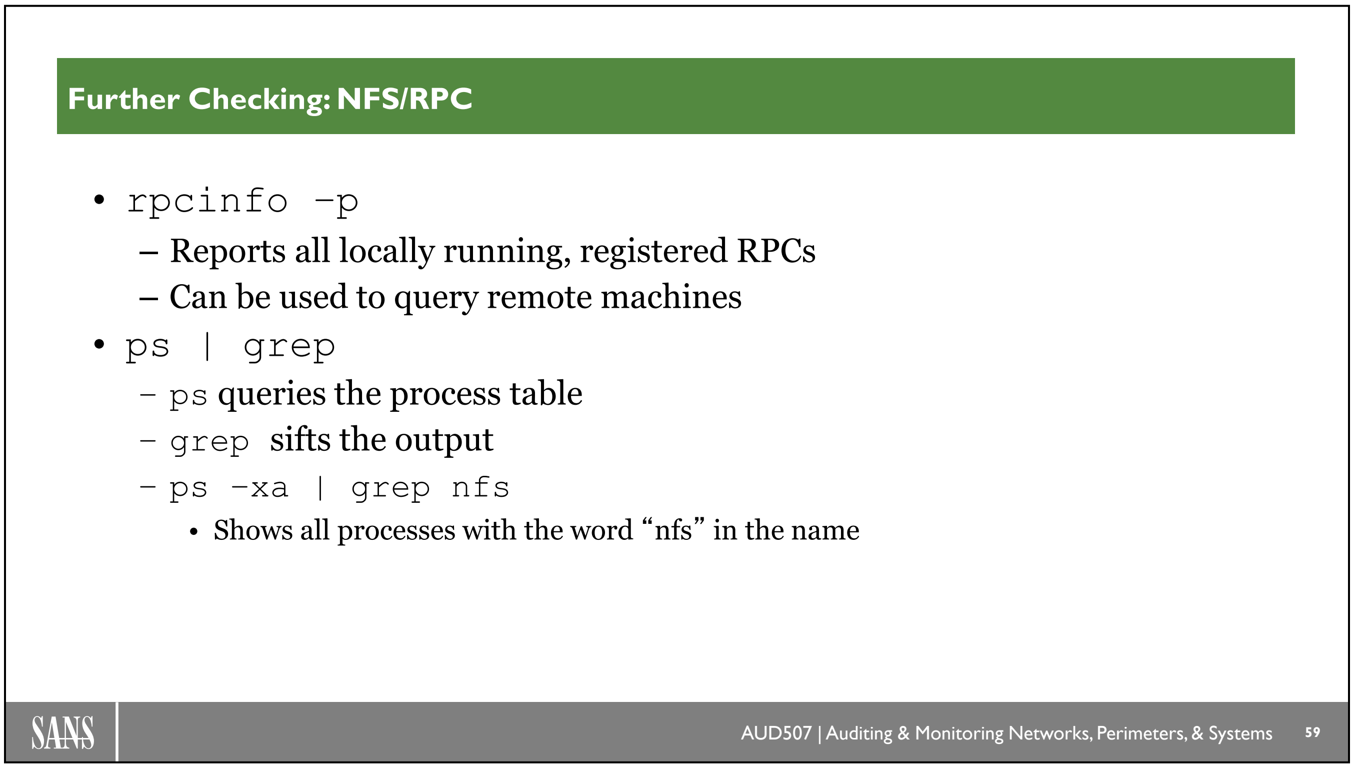
This gives us a very simple rule of thumb. If a remote filesystem is being mounted, we should see an entry that has either an IP address or a hostname in it. That same entry should also have an @ sign or a : that is used to separate the credentials and/or hostname from the actual path.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Further Checking: NFS/RPC

- `rpcinfo -p`
  - Reports all locally running, registered RPCs
  - Can be used to query remote machines
- `ps | grep`
  - `ps` queries the process table
  - `grep` sifts the output
  - `ps -xa | grep nfs`
    - Shows all processes with the word "nfs" in the name

On the local host itself, there are some other tricks we can do to determine whether there are RPC programs running or whether NFS is running. The easiest is to simply use the 'rpcinfo' utility. This program is used to query the portmapper directly and will return, among other things, a complete list of all currently running and registered RPC programs. We can even run this against a remote host by entering something like:

```
rpcinfo -p remoterpcs.mydomain.com
```

It is possible, however, for an RPC program to be running but not registered with the portmapper. The most common way that this happens is that either the portmapper is restarted at some point or the RPC service was started before the portmapper was started. In either of these cases, the portmapper will likely not have the information we seek. In fact, the portmapper might not even be running!

To find these services under these circumstances will require that we have done the research necessary to audit the particular brand of UNIX we are looking at. We will search the process table using the 'ps' command and try to identify programs that could possibly be RPC programs. For instance, we know that nfsd, statd, lockd, and mountd are all related to NFS. We can search for them like this:

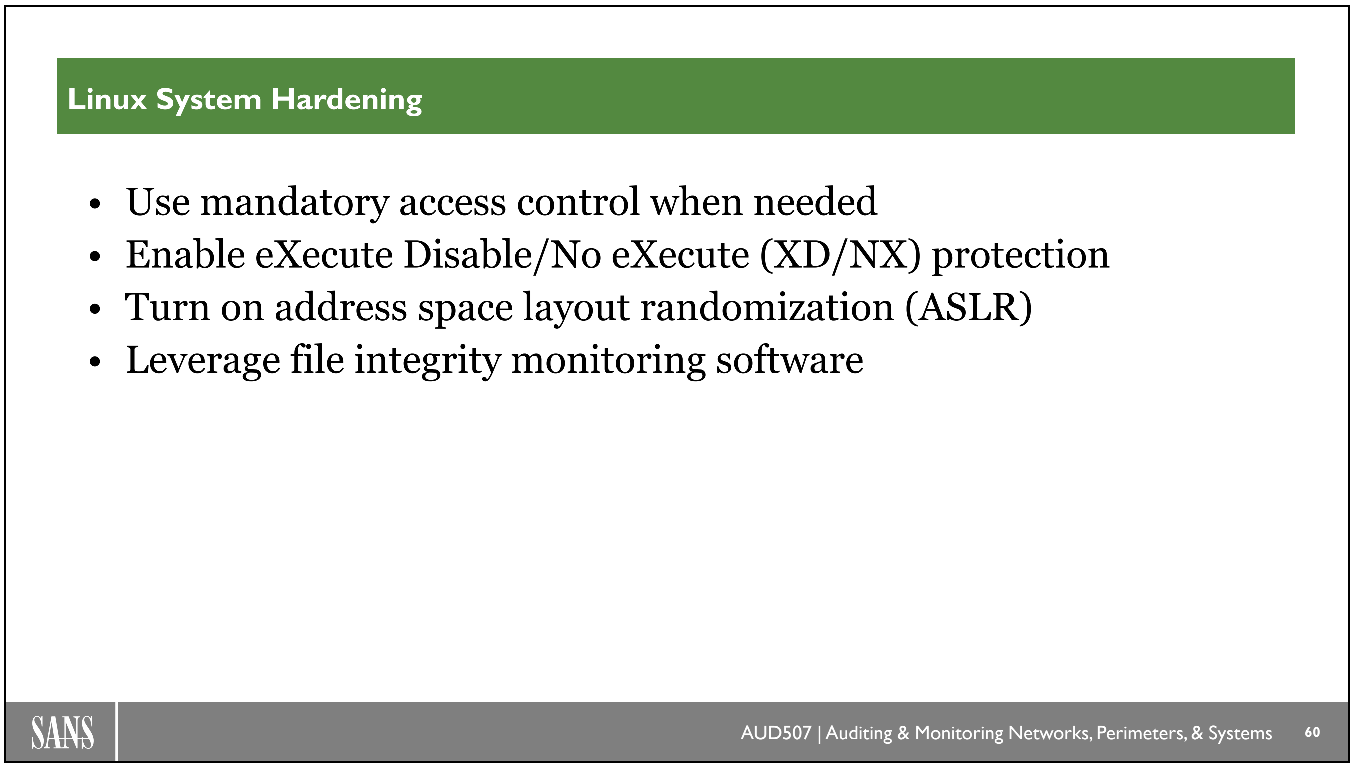
```
ps -eax | grep '(nfs)|(lockd)|(statd)|(mountd)|(rpc)'
```

Notice that we added the letters "rpc" to the end. On many systems, RPC programs will actually begin with "rpc.", allowing us to identify them more easily using this method.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Linux System Hardening

- Use mandatory access control when needed
- Enable eXecute Disable/No eXecute (XD/NX) protection
- Turn on address space layout randomization (ASLR)
- Leverage file integrity monitoring software

This slide lists some system hardening steps which can be used on most Linux systems. Mandatory access control (MAC) systems may be overkill for your systems, but when appropriate, they provide a good level of protection against the processes of one user interfering with the resources used by others. Execution prevention and address space layout randomization are used to prevent common attacks against poorly-coded applications.

Each is discussed more fully on the slides which follow.

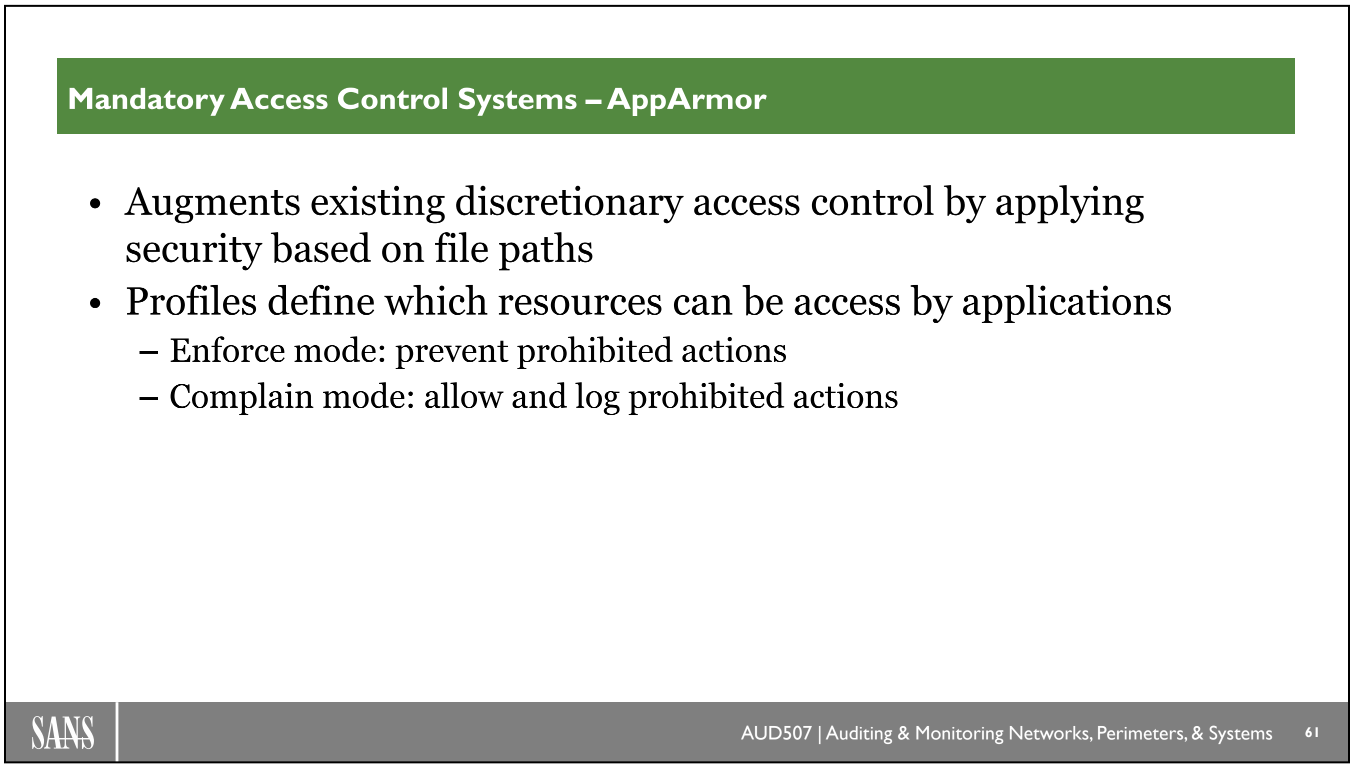
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Mandatory Access Control Systems – AppArmor

- Augments existing discretionary access control by applying security based on file paths
- Profiles define which resources can be access by applications
  - Enforce mode: prevent prohibited actions
  - Complain mode: allow and log prohibited actions

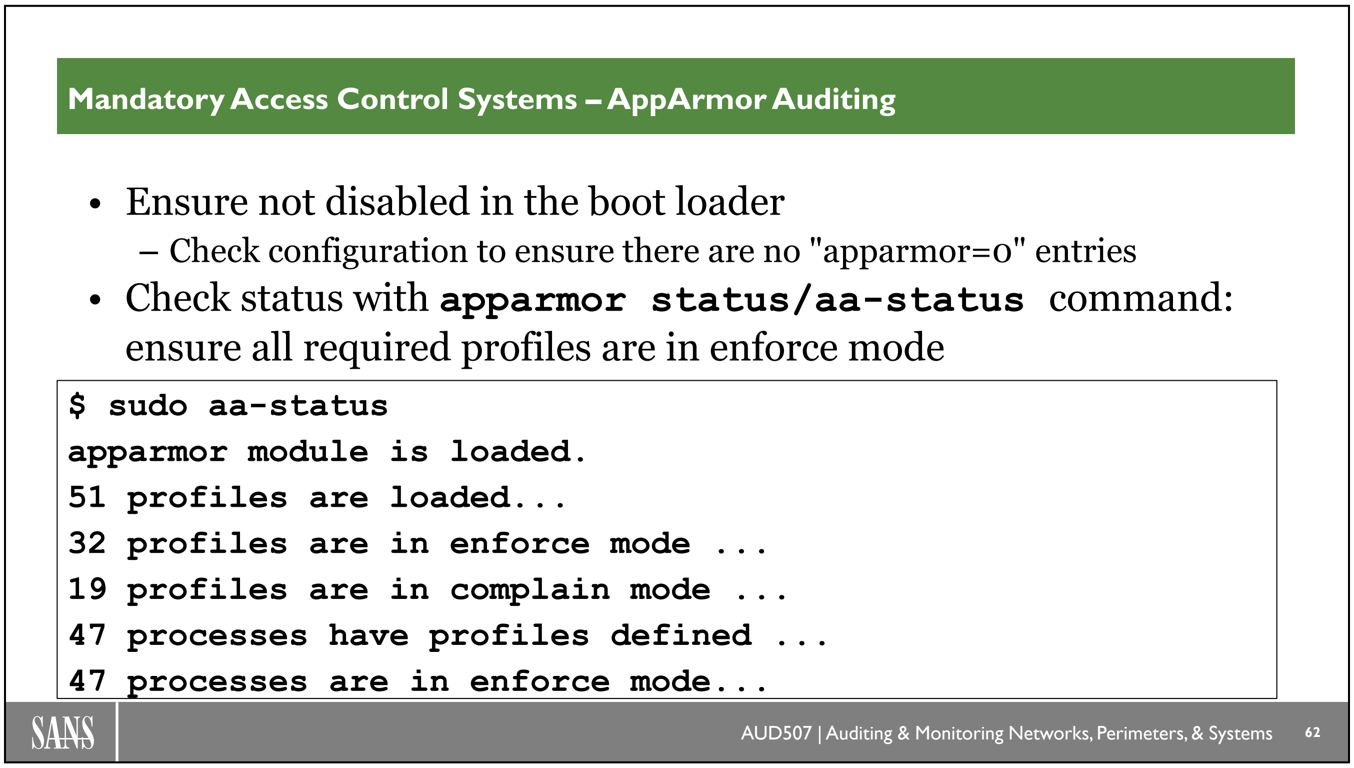
SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 61

AppArmor is a MAC system which applies controls based on file paths, rather than the more traditional model of using labels for subjects (like users and processed) and objects (like files and sockets). AppArmor profiles are used to define the access which should be allowed for specific programs. Each profile can be configured in either "enforce" mode, which will prevent any actions attempted that would violate the profile policy, or "complain" mode, which will allow the action but log it. Complain mode is often used for testing before placing a profile in the enforcement mode.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Mandatory Access Control Systems – AppArmor Auditing

- Ensure not disabled in the boot loader
  - Check configuration to ensure there are no "apparmor=0" entries
- Check status with **`apparmor status/aa-status`** command: ensure all required profiles are in enforce mode

```
$ sudo aa-status
apparmor module is loaded.
51 profiles are loaded...
32 profiles are in enforce mode ...
19 profiles are in complain mode ...
47 processes have profiles defined ...
47 processes are in enforce mode...
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 62

To validate that AppArmor has not been disabled in the boot loader configuration, search the configuration files to ensure that the string "apparmor=0" does NOT appear in the settings.

To check the status of the AppArmor system and get a list of profiles in the enforce and complain modes, run the **`apparmor status`** (or **`aa-status`**) command. Not only will this command list the current profiles and their statuses, it will also show you which applications are being controlled by the active profiles. The screenshot in this slide was truncate, as the command returns a list of every profile or process in each mode.

On Ubuntu systems, if you'd like to get the count of profiles in a certain mode, run aa-status with one of these flags (only one allowed per command line): --complaining, --enforced, --enabled, or --profiled. See the help output for aa-status for more information.

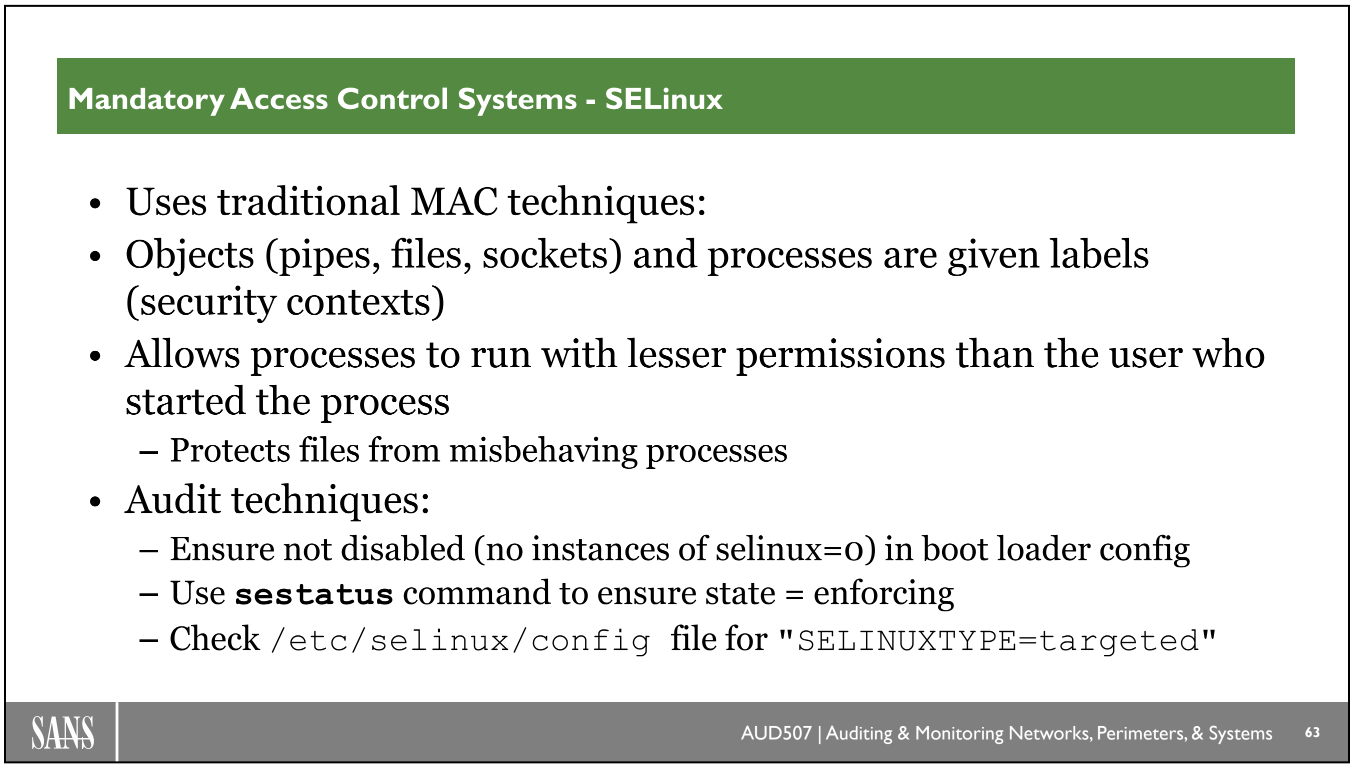
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Mandatory Access Control Systems - SELinux

- Uses traditional MAC techniques:
- Objects (pipes, files, sockets) and processes are given labels (security contexts)
- Allows processes to run with lesser permissions than the user who started the process
  - Protects files from misbehaving processes
- Audit techniques:
  - Ensure not disabled (no instances of selinux=0) in boot loader config
  - Use **`sestatus`** command to ensure state = enforcing
  - Check `/etc/selinux/config` file for `"SELINUXTYPE=targeted"`

SELinux is a much more traditional MAC system, in that it uses labels for processes and objects. For an action to happen, it must be allowed by the normal discretionary access controls (i.e., file permissions) in the OS, AND the MAC rules. A violation of either will cause the action to be denied.

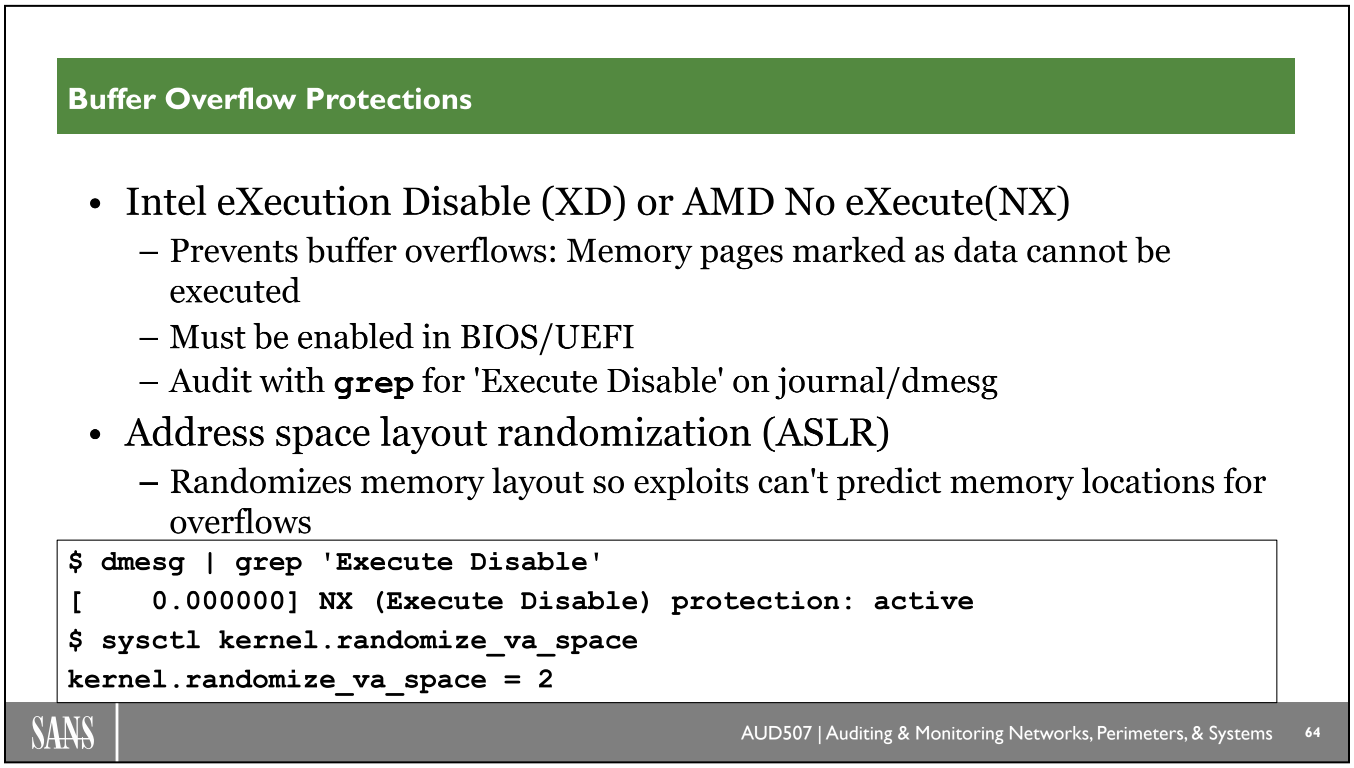
SELinux has the concept of a transition, which means that an application can be made to run with even lower permissions on objects than the user who started the process. This adds an extra layer of protection for objects.

To validate the settings, ensure that SELinux is not disabled in the boot loader, use the sestatus command to ensure the current state "enforcing," and verify that the configuration file for SELinux has the line "SELINUXTYPE=targeted" in it.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Buffer Overflow Protections

- Intel eXecution Disable (XD) or AMD No eXecute(NX)
  - Prevents buffer overflows: Memory pages marked as data cannot be executed
  - Must be enabled in BIOS/UEFI
  - Audit with **grep** for 'Execute Disable' on journal/dmesg
- Address space layout randomization (ASLR)
  - Randomizes memory layout so exploits can't predict memory locations for overflows

```
$ dmesg | grep 'Execute Disable'
[    0.000000] NX (Execute Disable) protection: active
$ sysctl kernel.randomize_va_space
kernel.randomize_va_space = 2
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 64

The execute disable feature for processors prevents memory pages marked for data use from being executed. This effectively mitigates many buffer overflow attacks, since they tend to overwrite memory reserved for data with their malicious code. This feature must be enabled in the BIOS or UEFI system before it can be used. Also, on 32-bit systems, the physical address extension (PAE) must be enabled in the running kernel for the feature to work (verify this capability with a grep for 'pae' in /proc/cpuinfo). To audit for this setting, verify that either the systemd journal or dmesg report "protection: active" for Execute Disable.

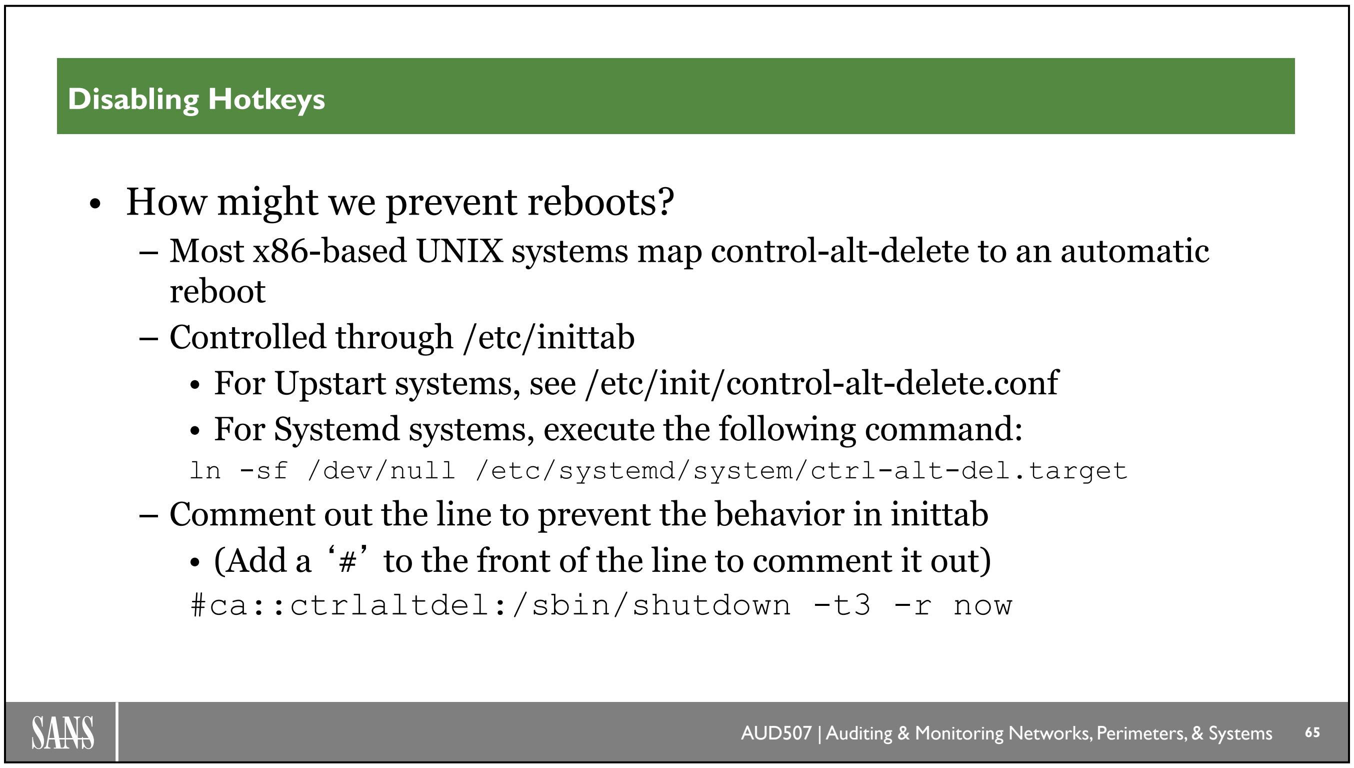
Address space layout randomization seeks to prevent buffer overflows by making it more difficult to predict the location of any given data in memory. This can make memory page exploits much harder to perform. Validate it by checking that the kernel.randomize_va_space sysctl value is set to '2.'

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Disabling Hotkeys

- How might we prevent reboots?
  - Most x86-based UNIX systems map control-alt-delete to an automatic reboot
  - Controlled through /etc/inittab
    - For Upstart systems, see /etc/init/control-alt-delete.conf
    - For Systemd systems, execute the following command:

```
ln -sf /dev/null /etc/systemd/system/ctrl-alt-del.target
```

  - Comment out the line to prevent the behavior in inittab
    - (Add a '#' to the front of the line to comment it out)

```
#ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

It would also be good to disable hotkeys that perform automatic reboots. This won't prevent someone from pulling the plug, but hopefully, our data center is just a bit more secure than average, and servers are kept within locked cages and consoles are locked with passwords. Even in this scenario, someone could press Control-Alt-Delete to initiate a reboot on most x86-based UNIX systems, but this can be controlled as well.

The /etc/inittab file usually will contain a line that begins with the letters "ca" that is used to define what actions to take when these keys are hit. If we remove or comment out this line, it will no longer be possible to initiate a reboot simply by pressing Control-Alt-Delete. Instead, it will now be necessary for root to log in and manually run the reboot command.

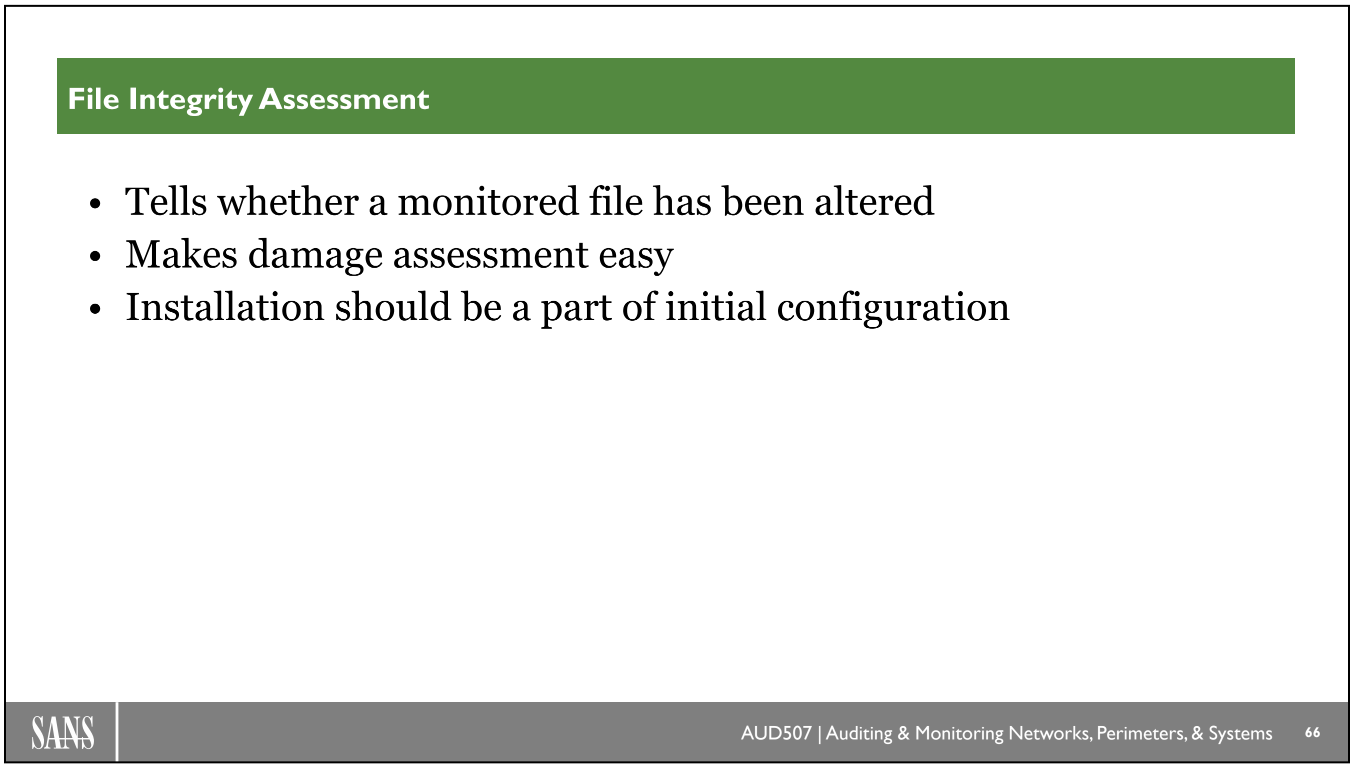
Upstart systems control this using a configuration file, /etc/init/control-alt-delete.conf. Systemd, on the other hand, requires a configuration file be created to disable this facility.

The command in the slide related to Systemd is the "Link" command. This link command, as opposed to the "ld" command, is used to create links within the filesystem. In this case, a symbolic (the –s) link of type "File" (the –f) is being created, making /etc/systemd/system/ctrl-alt-del.target an alias for /dev/null (otherwise known as the bit-bucket).

August 10, 2021

August 10, 2021

© SANS Institute 2021



File Integrity Assessment

- Tells whether a monitored file has been altered
- Makes damage assessment easy
- Installation should be a part of initial configuration

Make sure that any system deployed has the ability to detect changes in important system files. Something as simple as adding a "++" to the /.rhosts file allowed Kevin Mitnick to render Shimomura's system completely defenseless.

Of course, being able to detect that a file has changed means that the system was compromised, and that's kind of late in the game. But we would still rather know 12 hours after the compromise than not to know at all.

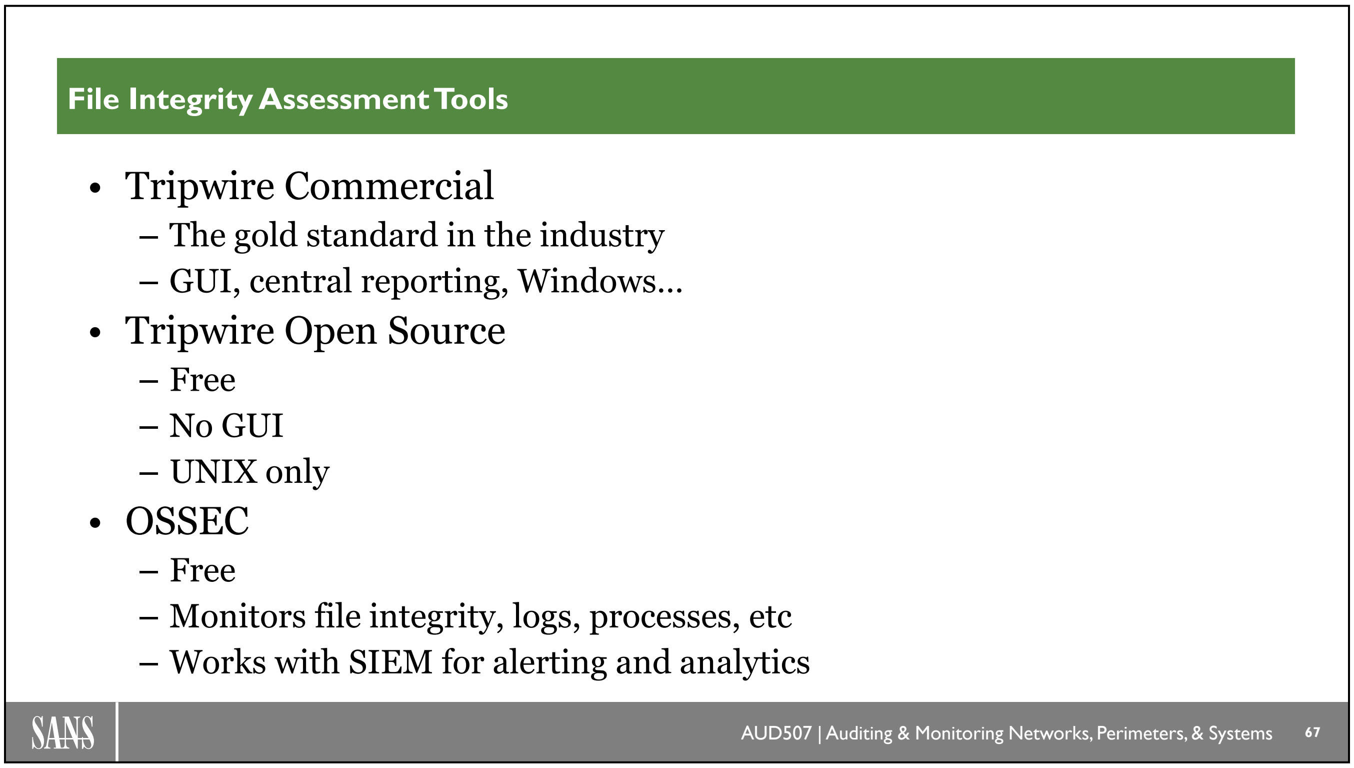
The ability to detect changes in the files of the filesystem also makes recovering from an incident much easier. In this case, you often are able to assess exactly which files were changed by the attacker. This means that recovery can consist of replacing the hacked files (and securing the system) as opposed to a complete reinstall and restore from backups.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



File Integrity Assessment Tools

- Tripwire Commercial
  - The gold standard in the industry
  - GUI, central reporting, Windows…
- Tripwire Open Source
  - Free
  - No GUI
  - UNIX only
- OSSEC
  - Free
  - Monitors file integrity, logs, processes, etc
  - Works with SIEM for alerting and analytics

The installation of file integrity–testing software is an absolute must for any server-class system. The installation of this software should really be a part of the base configuration of the system.

There are lots of options in this market space for UNIX. You should push your organization toward Tripwire because it will allow you to create a single file integrity management infrastructure across your enterprise. Unfortunately, that's going to cost you some money.

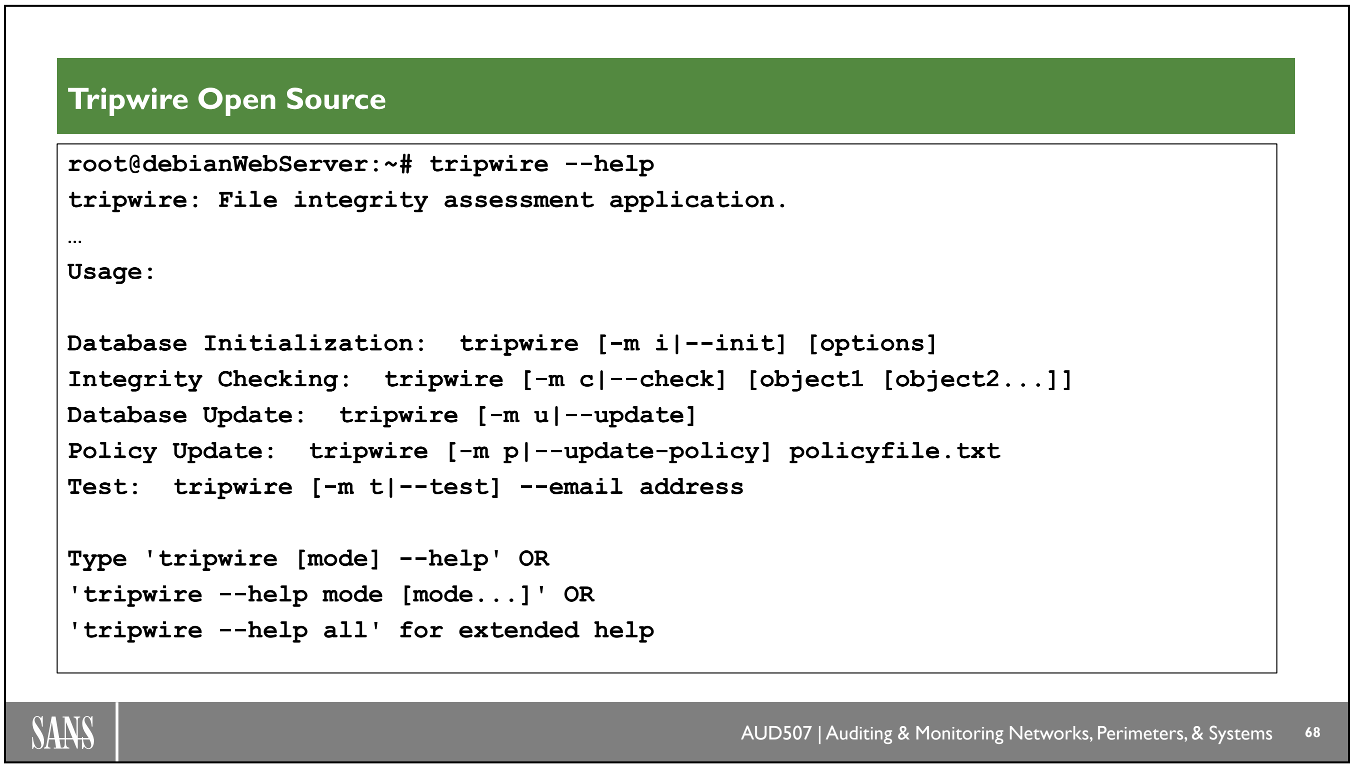
If money is tight, you might start by looking at the Open Source version of Tripwire. We look at an item or two that can cause headaches for administrators when first using this tool in a second.

Remember that OSSEC is built to be used with a SIEM and may be a good solution for enterprises who want to feed file integrity events into their SIEM.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Tripwire Open Source

```
root@debianWebServer:~# tripwire --help
tripwire: File integrity assessment application.
…
Usage:

Database Initialization:  tripwire [-m i|--init] [options]
Integrity Checking:  tripwire [-m c|--check] [object1 [object2...]]
Database Update:  tripwire [-m u|--update]
Policy Update:  tripwire [-m p|--update-policy] policyfile.txt
Test:  tripwire [-m t|--test] --email address

Type 'tripwire [mode] --help' OR
'tripwire --help mode [mode...]' OR
'tripwire --help all' for extended help
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 68

In the slide, we are picturing the Open Source version of Tripwire. This version has been written with Linux specifically in mind, though it will work on Solaris or BSD with a minimal amount of effort. The Open Source version is available from https://github.com/Tripwire/tripwire-open-source and is also free for use by the security community.

With all of these free options, you might wonder why you would want to purchase the commercial version of Tripwire! The answer is that the commercial version includes some fantastic reporting options in addition to being available for multiple platforms that are not supported on the Open Source side. These multiplatform versions also work cooperatively, giving you the opportunity to create integrated reports.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021

Tripwire Usage

- Build, test, and install
- Edit Tripwire configuration file
- Decide on a location for Tripwire files
- Initialize Tripwire database
- Copy database to secure media

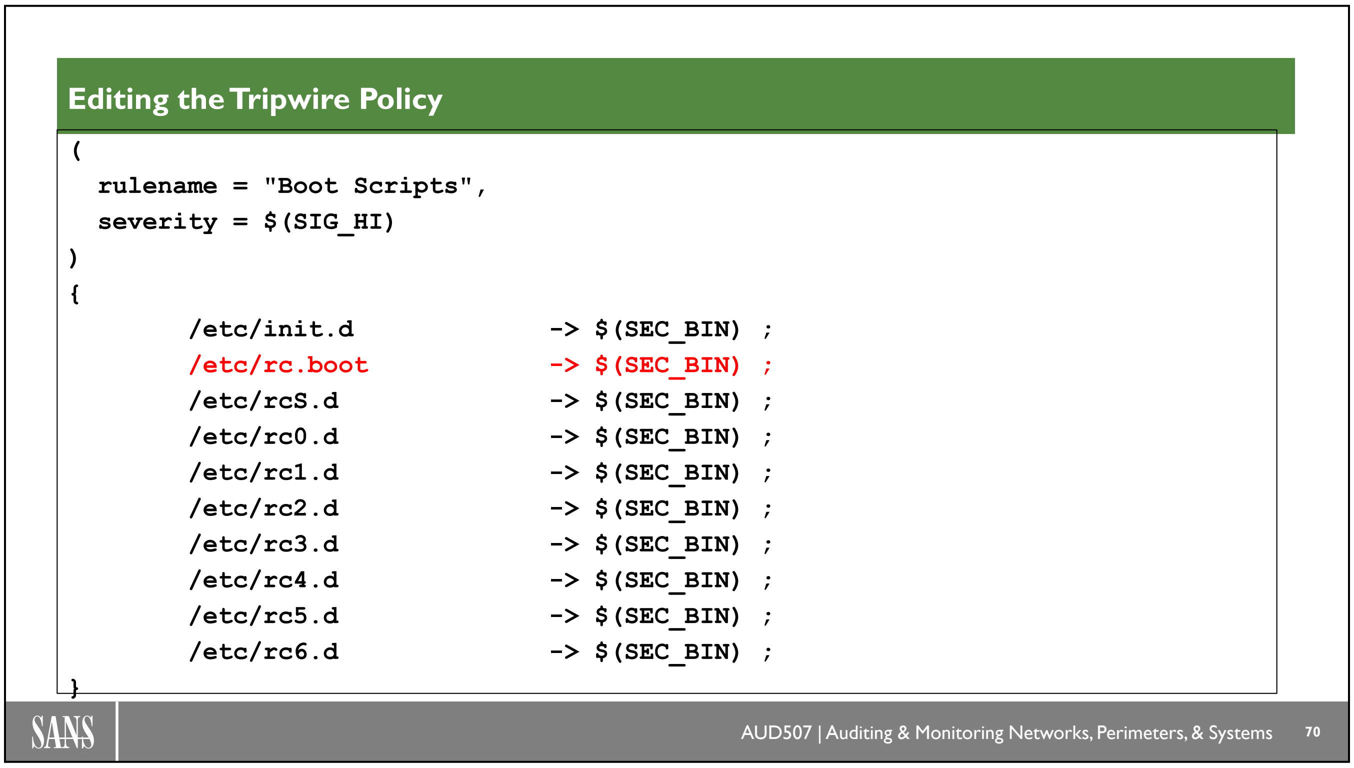
To put Tripwire into use, there are a few steps that you need to take. First, after deciding on which version you plan to use, you must build the source code into binaries. The Tripwire source code, if you choose to use the Academic Source Release (ASR) or the Open Source version, comes with detailed instructions on how to go about building Tripwire. If you purchase the commercial version, you will not need to take these steps. You will simply need to install the software.

After the software is installed, you must edit the Tripwire configuration file to customize the tool for your system and decide what you would like to baseline. There are a number of sample configuration files that come with Tripwire to get you started. Next, you decide on a location for the Tripwire database and configuration files and copy them to this location. At this point, you are ready to go so you initialize the Tripwire database. This will perform an initial assessment of the system and generate fingerprints for everything specified in the configuration file. Finally, you must move the database that is created to secure media to prevent tampering.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Editing the Tripwire Policy

```
(
  rulename = "Boot Scripts",
  severity = $(SIG_HI)
)
{
        /etc/init.d              -> $(SEC_BIN) ;
        /etc/rc.boot             -> $(SEC_BIN) ;
        /etc/rcS.d               -> $(SEC_BIN) ;
        /etc/rc0.d               -> $(SEC_BIN) ;
        /etc/rc1.d               -> $(SEC_BIN) ;
        /etc/rc2.d               -> $(SEC_BIN) ;
        /etc/rc3.d               -> $(SEC_BIN) ;
        /etc/rc4.d               -> $(SEC_BIN) ;
        /etc/rc5.d               -> $(SEC_BIN) ;
        /etc/rc6.d               -> $(SEC_BIN) ;
}
```

Editing the Tripwire policy file is pretty simple once you understand the format. We're not going to spend time discussing each of the individual configuration options for the policy file. The key is to identify portions of the system that should and should not change and then enumerate them in this file. After this is done, you can create a pretty good baseline of the system, looking for changes in everything from modification times and sizes, all the way to the inode used by a file.

The administrator really ought to go over this file carefully before initializing the database, but there are thousands of files on any given UNIX system. More typically, as mentioned on the last page, he will run the initialization and then come and clean it up by editing this file. He's headed for some frustration, though, and we just want to prepare you for that. If you are introducing this tool to an administrator, it would be a really good idea to explain what's going to happen next.

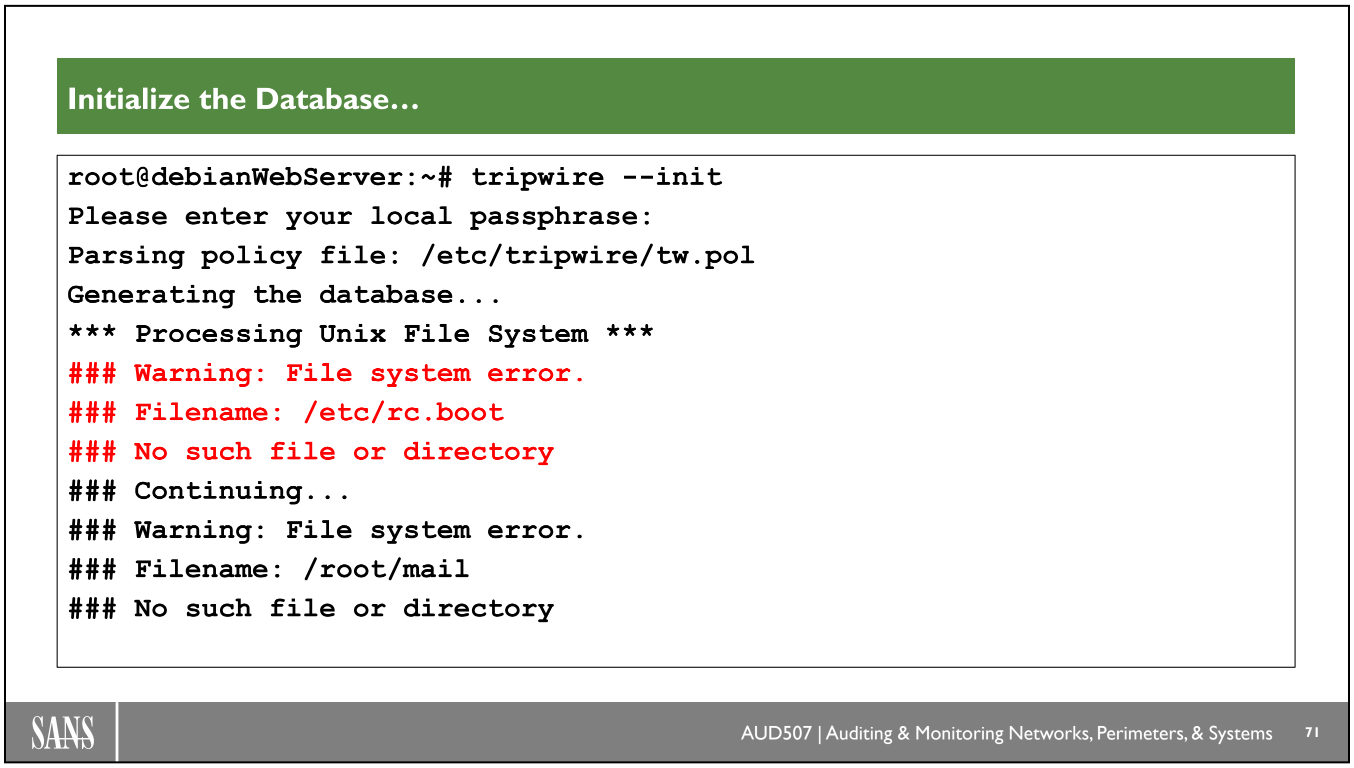
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Initialize the Database…

```
root@debianWebServer:~# tripwire --init
Please enter your local passphrase:
Parsing policy file: /etc/tripwire/tw.pol
Generating the database...
*** Processing Unix File System ***
### Warning: File system error.
### Filename: /etc/rc.boot
### No such file or directory
### Continuing...
### Warning: File system error.
### Filename: /root/mail
### No such file or directory
```

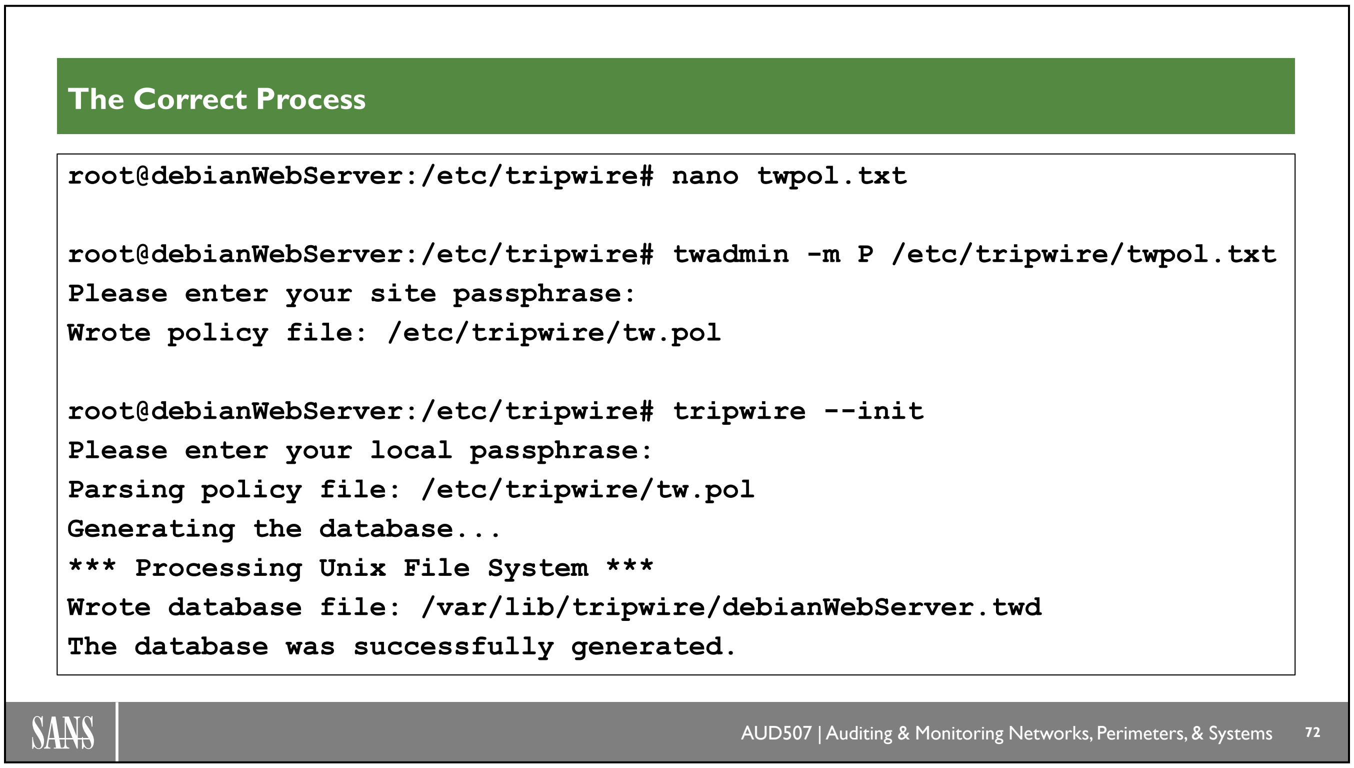
Once installed, the Open Source and Commercial versions must create the fingerprint database before anything else is done (notice that error messages are being generated). This is a sign that the tool has not been correctly configured. Unfortunately, many administrators will simply choose to use the defaults, which can lead to an issue down the road. We'd like to find that we have nice, clean Tripwire reports. This requires that the administrators take the time to edit the configuration, eliminating references to things that do not exist on the system in question, or changing the path for things that reside in different locations.

Remember that every UNIX system is somewhat unique. The Tripwire system comes with a default configuration, but it must be customized for your specific system. It is typical for the administrator to allow it to perform the initialization and then go back to fix all of the warnings. He does this by editing the twpol.txt file depicted in the previous slide.

August 10, 2021

August 10, 2021

© SANS Institute 2021



The Correct Process

```
root@debianWebServer:/etc/tripwire# nano twpol.txt

root@debianWebServer:/etc/tripwire# twadmin -m P /etc/tripwire/twpol.txt
Please enter your site passphrase:
Wrote policy file: /etc/tripwire/tw.pol

root@debianWebServer:/etc/tripwire# tripwire --init
Please enter your local passphrase:
Parsing policy file: /etc/tripwire/tw.pol
Generating the database...
*** Processing Unix File System ***
Wrote database file: /var/lib/tripwire/debianWebServer.twd
The database was successfully generated.
```

In this slide, we present the steps that should normally be undertaken by an administrator to properly configure Tripwire and initialize the database. It is critical to remember that that these steps should result in NO ERRORS when everything has been configured correctly.

August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021

Copy to Secure Media

- Database file holds fingerprints
- Policy file tells what to baseline
- Attackers will modify these files
- Change control
- Solution:
  - Copy offline
  - Burn to DVD
  - MD5sum them

After the baseline is created, the most important step is to safeguard the fingerprint database. Many sites will go as far as installing and configuring Tripwire and then fail to safeguard the fingerprints, thus rendering Tripwire "mostly harmless" to the attacker.

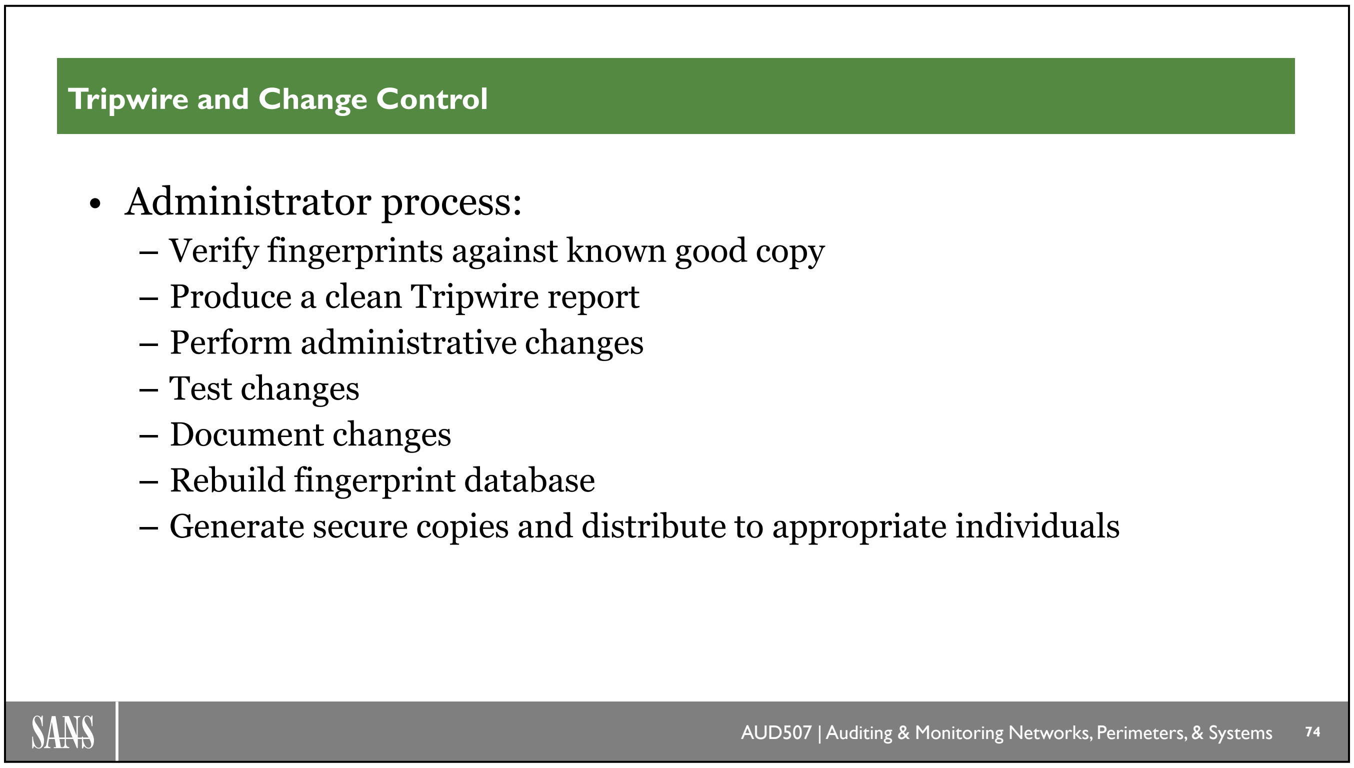
One of the first things that an attacker does after establishing a beachhead on your system is to begin to cover his tracks. This means that he will begin deleting log entries, changing logging options, and, yes, updating the Tripwire database if he realizes it's there.

Besides attackers, we have the continuing problem of enforcing and tracking changes to our systems. Let's be honest: System administrators are not the best when it comes to documenting changes that they make to systems. Using a tool like Tripwire, however, we now create a control that serves multiple purposes. Not only is Tripwire a security control in that it can alert us to changes, but it is also an audit control for our change control process.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Tripwire and Change Control

- Administrator process:
  - Verify fingerprints against known good copy
  - Produce a clean Tripwire report
  - Perform administrative changes
  - Test changes
  - Document changes
  - Rebuild fingerprint database
  - Generate secure copies and distribute to appropriate individuals

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 74

The process that your administrators should follow is a pretty simple one and does not require an onerous amount of labor. In fact, it quite likely incorporates security procedures that are not followed today. The idea is that the administrator must first verify that the system has not changed since the last baseline was run by checking the file fingerprints against the last known good copy of the database. After this is completed, the administrator can go ahead and do whatever patching or installation work needs to be done and document those changes appropriately. The next step is to rebuild the fingerprint database and to create secure copies of it to be distributed to the responsible individuals.

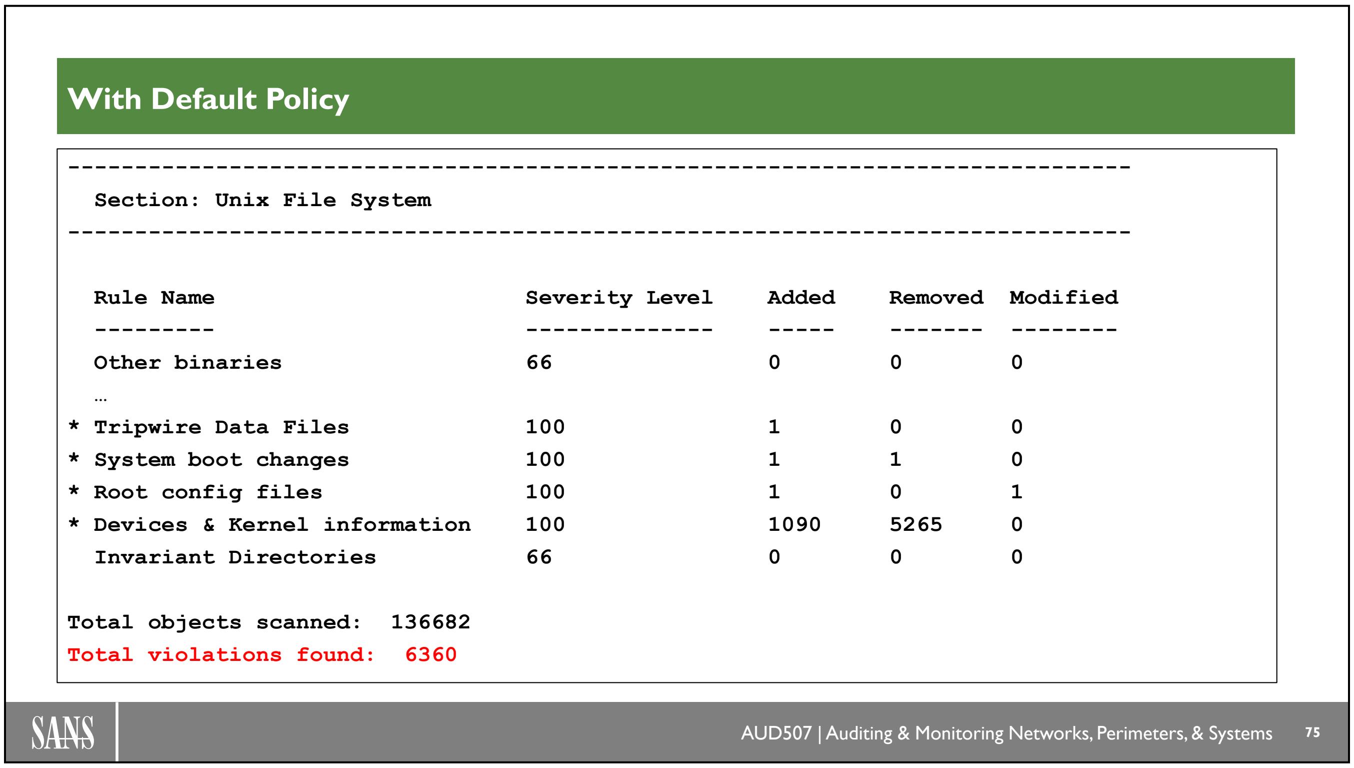
In an environment with hundreds (or more) of servers, the process would likely not send a copy of the database to the auditor after every change, but there would be secure copies kept in the data center vault, for instance. The auditor will always have his own last known good baseline and can verify change control process documentation by comparing the report from his known good copy against the running system, and then simply update his own baseline.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



With Default Policy

```
-------------------------------------------------------------------------------
  Section: Unix File System
-------------------------------------------------------------------------------

  Rule Name                         Severity Level    Added    Removed  Modified
  ---------                         --------------    -----    -------  --------
  Other binaries                    66                0        0        0
  …
* Tripwire Data Files               100               1        0        0
* System boot changes               100               1        1        0
* Root config files                 100               1        0        1
* Devices & Kernel information      100               1090     5265     0
  Invariant Directories             66                0        0        0

Total objects scanned:  136682
Total violations found:  6360
```

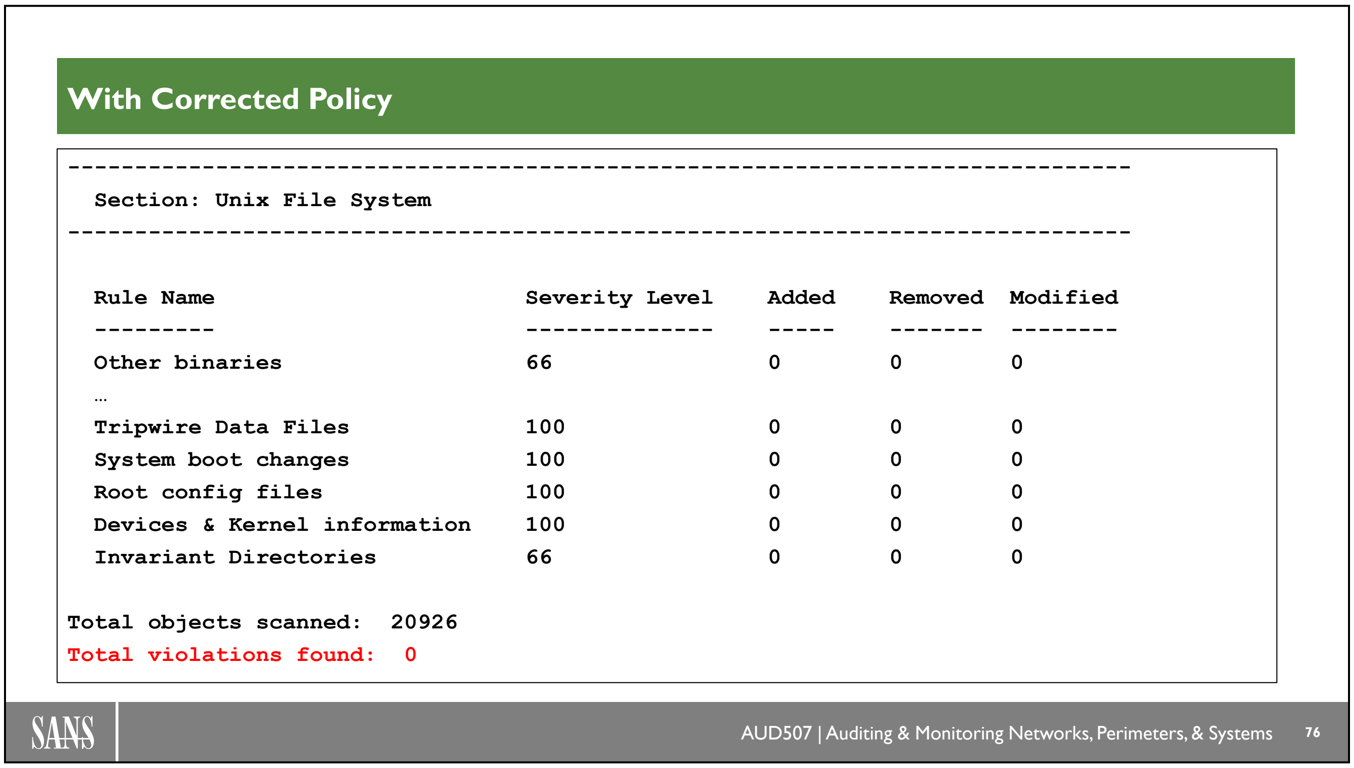
SANS

In this screenshot you see a report run using an installation of Tripwire which has not been properly configured. The administrator should never accept a report with any exceptions or errors as “normal.”

August 10, 2021

August 10, 2021

© SANS Institute 2021



With Corrected Policy

```
--------------------------------------------------------------------------------
  Section: Unix File System
--------------------------------------------------------------------------------

  Rule Name                         Severity Level    Added    Removed  Modified
  ---------                         --------------    -----    -------  --------
  Other binaries                    66                0        0        0
  …
  Tripwire Data Files               100               0        0        0
  System boot changes               100               0        0        0
  Root config files                 100               0        0        0
  Devices & Kernel information      100               0        0        0
  Invariant Directories             66                0        0        0

Total objects scanned:  20926
Total violations found:  0
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 76

In this example, you see the results of a check run on a properly configured system. The goal should always be to have a total of zero errors or violations to prove that the system is in a known-good state.

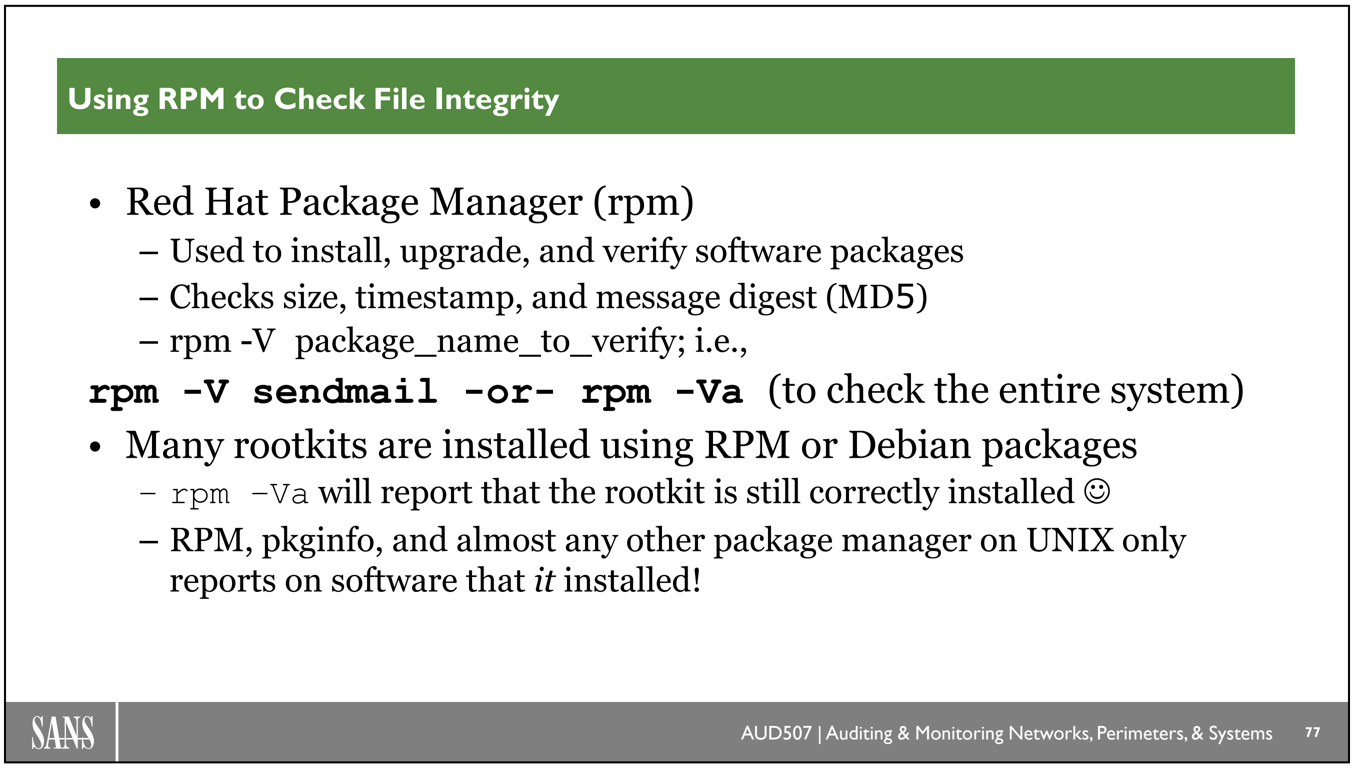
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Using RPM to Check File Integrity

- Red Hat Package Manager (rpm)
  - Used to install, upgrade, and verify software packages
  - Checks size, timestamp, and message digest (MD5)
  - rpm -V package_name_to_verify; i.e.,

`rpm -V sendmail -or- rpm -Va` (to check the entire system)

- Many rootkits are installed using RPM or Debian packages
  - `rpm -Va` will report that the rootkit is still correctly installed ☺
  - RPM, pkginfo, and almost any other package manager on UNIX only reports on software that *it* installed!

SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 77

If you begin the conversation about file integrity testing with the administrator, be prepared for another objection. He might say, "Oh, yes, I know about that. But we don't need it because we're using [X] instead!" The X here is some type of package management product like Apt, Yast, RPM, or some other similar installation tool.

Although these tools have done a great deal to simplify administration of UNIX systems and the installation of software, they are *not* file integrity–testing tools! They certainly have some hashing features built in, but the purpose of these tools is to simply inspect installed software for the purpose of repairing installations or applying patches.

Although these tools are valuable, they are capable only of reporting on the integrity of files that they themselves installed! They will not be able to tell you whether there are new files, or whether there were files modified that were not a part of a package installed or maintained through the package manager that you are using!

This is a pretty important distinction between a tool like Tripwire and the RPM package manager. If you want to verify a package, by all means use RPM. If you are trying to verify a system, Tripwire (or something like it) is the correct tool for the job.

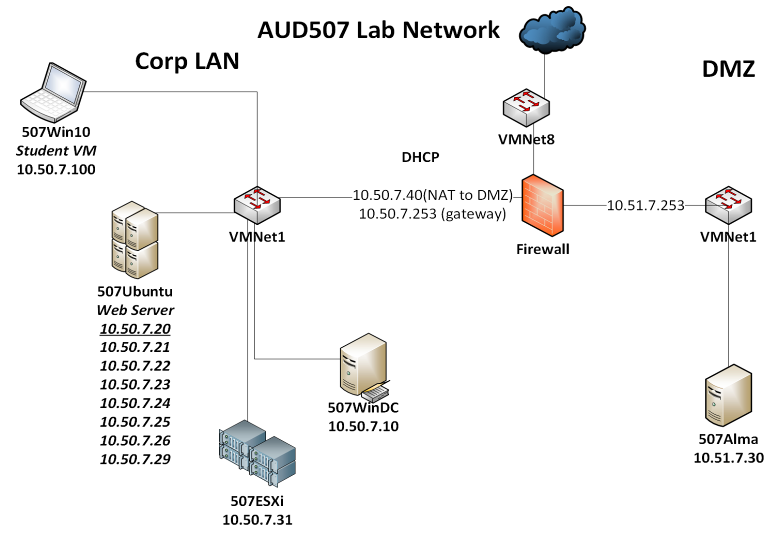
Realize, too, that package management tools have become the primary way for rootkits to be installed onto UNIX systems by attackers. This means that if you are relying on a tool such as these to validate your system, it will happily tell you that everything is just fine because the rootkit is still correctly installed.

August 10, 2021

August 10, 2021

© SANS Institute 2021

Exercise 3.2: System Information, Permissions, and File Integrity



Please open your Workbook to Exercise 3.2: System Information, Permissions, and File Integrity, or pick up with wherever you left off in the exercises.

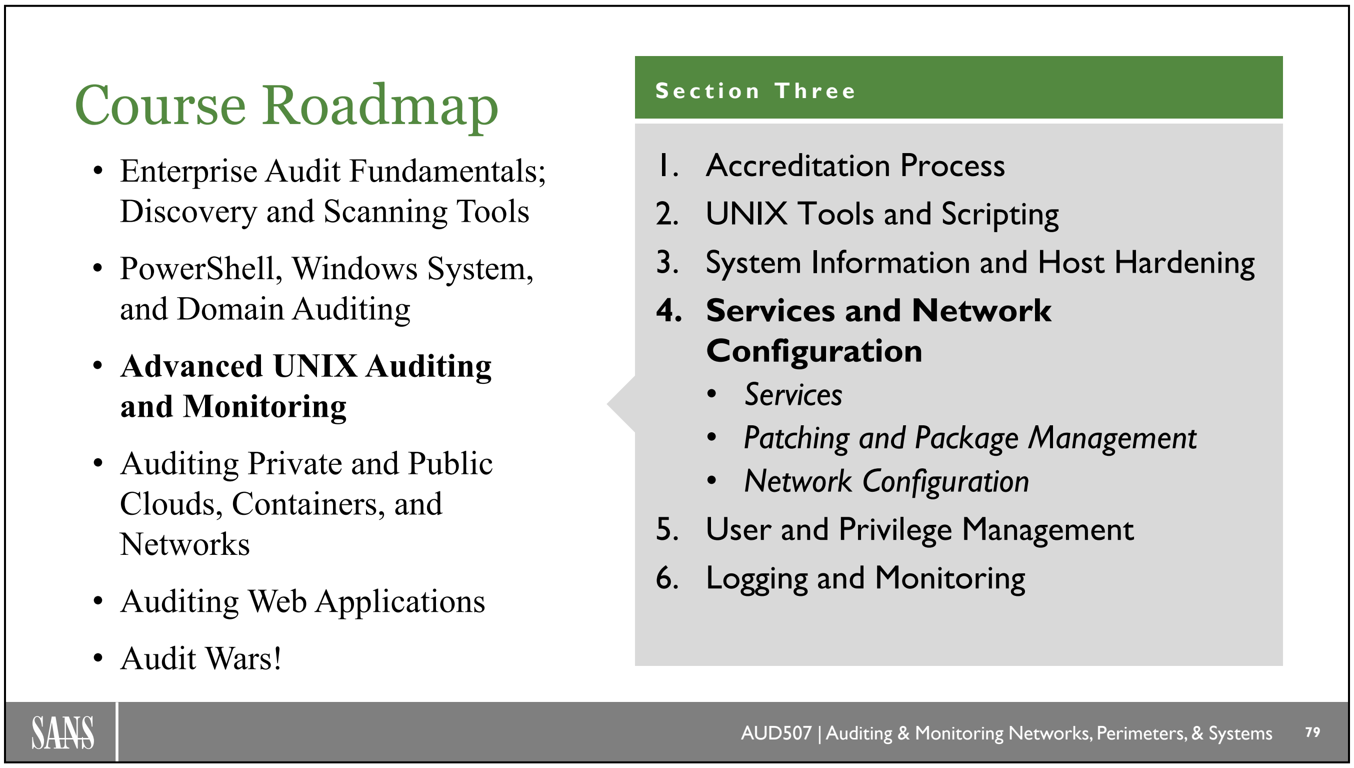
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Course Roadmap

- Enterprise Audit Fundamentals; Discovery and Scanning Tools
- PowerShell, Windows System, and Domain Auditing
- **Advanced UNIX Auditing and Monitoring**
- Auditing Private and Public Clouds, Containers, and Networks
- Auditing Web Applications
- Audit Wars!

Section Three

1. Accreditation Process
2. UNIX Tools and Scripting
3. System Information and Host Hardening
4. **Services and Network Configuration**
    - *Services*
    - *Patching and Package Management*
    - *Network Configuration*
5. User and Privilege Management
6. Logging and Monitoring

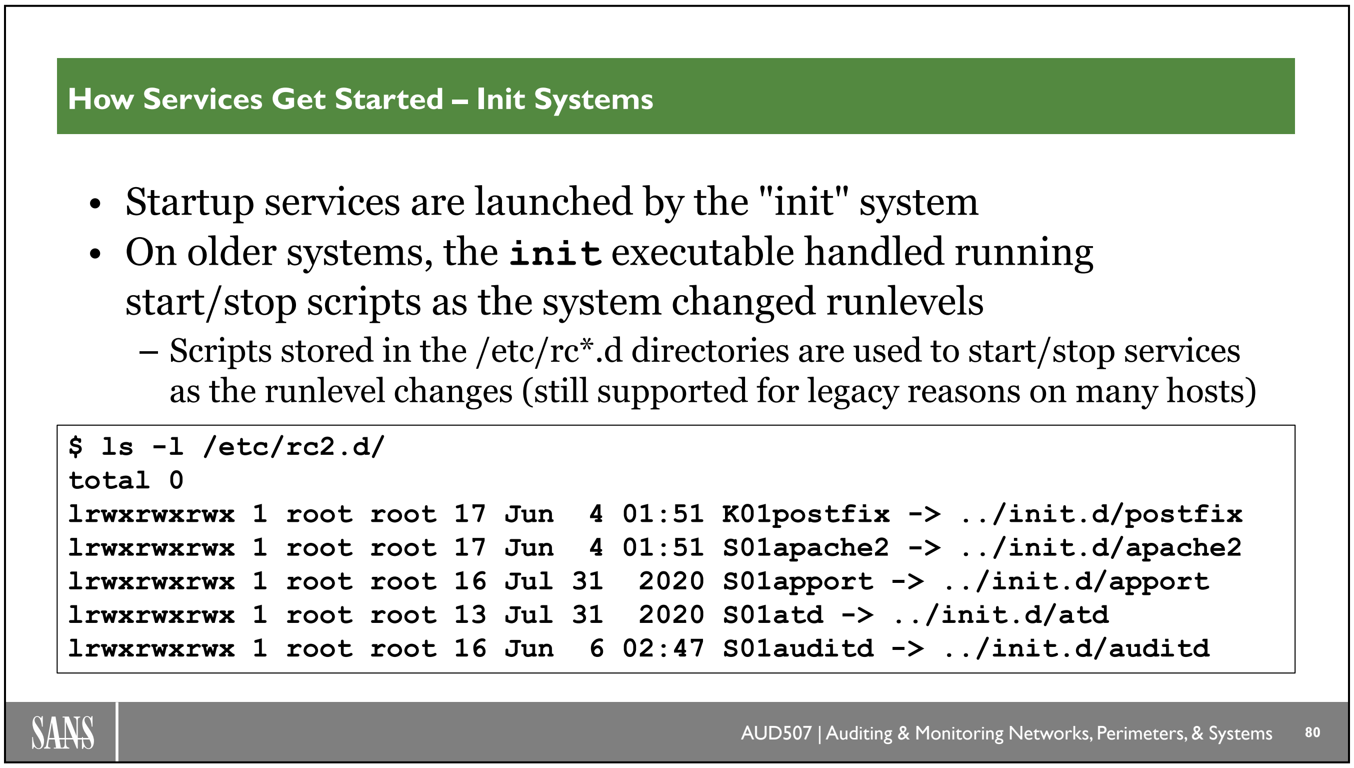
SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 79

In this section, we discuss Linux service configuration, OS and application patching, and hardening the network configuration for a host.

August 10, 2021

August 10, 2021

© SANS Institute 2021



How Services Get Started – Init Systems

- Startup services are launched by the "init" system
- On older systems, the **`init`** executable handled running start/stop scripts as the system changed runlevels
  - Scripts stored in the /etc/rc*.d directories are used to start/stop services as the runlevel changes (still supported for legacy reasons on many hosts)

```
$ ls -l /etc/rc2.d/
total 0
lrwxrwxrwx 1 root root 17 Jun  4 01:51 K01postfix -> ../init.d/postfix
lrwxrwxrwx 1 root root 17 Jun  4 01:51 S01apache2 -> ../init.d/apache2
lrwxrwxrwx 1 root root 16 Jul 31  2020 S01apport -> ../init.d/apport
lrwxrwxrwx 1 root root 13 Jul 31  2020 S01atd -> ../init.d/atd
lrwxrwxrwx 1 root root 16 Jun  6 02:47 S01auditd -> ../init.d/auditd
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 80

In the *nix dark ages, prior to around 2006, the init process handled "runlevel" changes for our hosts. Runlevels represented different states in which the host might find itself (initial boot, single-user mode, multi-user mode, GUI running, etc.). As the runlevel changed during boot and shutdown processes, init had the job of running start and kill scripts for various tools and services. The scripts were saved in directories named after the runlevel and started with a number (to dictate execution order) and a capital S or K to denote if the script should be run in "start" or "kill" mode. The init process was very predictable, but SLOW, because the scripts were run synchronously. Each script would start only after the previous had ended, meaning that a slow-running script could greatly increase system boot time.

Even the Ubuntu 20.04 LTS (released in April of 2020) host in the screenshot still has vestiges of this older initialization process.

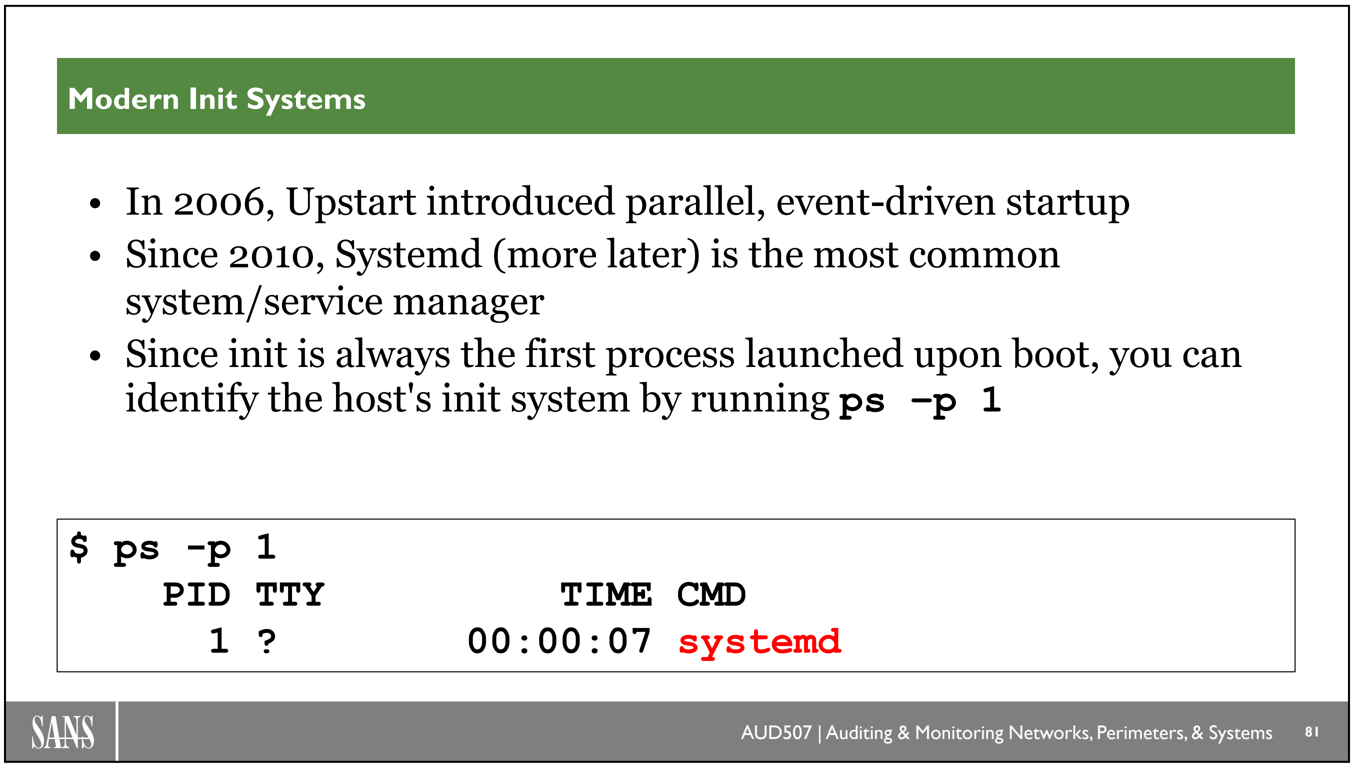
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Modern Init Systems

- In 2006, Upstart introduced parallel, event-driven startup
- Since 2010, Systemd (more later) is the most common system/service manager
- Since init is always the first process launched upon boot, you can identify the host's init system by running **`ps -p 1`**

```
$ ps -p 1
    PID TTY          TIME CMD
      1 ?        00:00:07 systemd
```

Introduced in 2006, Upstart was used as an event-driven initialization tool for many Linuxes. By using events like "net-device-up" and desktop-session-start," Upstart could run in parallel all the scripts which depended on the event. This made the system appear to boot much faster, even if some long-running scripts were not yet completed when the user saw the login prompt.

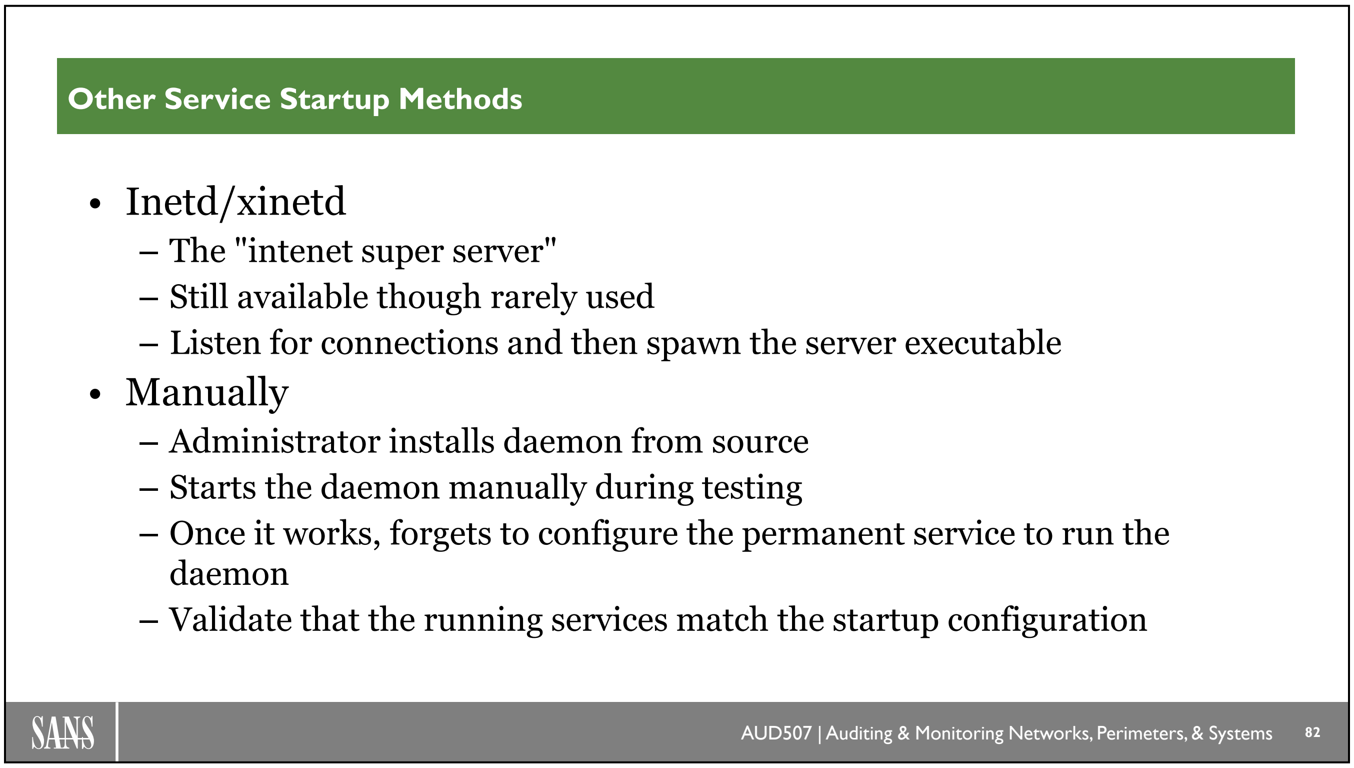
Since about 2010 most Linux systems use Systemd (covered just a bit later) as the main startup system.

You can usually tell which init system the host is using by checking process information for process id (pid) 1 with the command **`ps -p 1`**.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Other Service Startup Methods

- Inetd/xinetd
  - The "intenet super server"
  - Still available though rarely used
  - Listen for connections and then spawn the server executable
- Manually
  - Administrator installs daemon from source
  - Starts the daemon manually during testing
  - Once it works, forgets to configure the permanent service to run the daemon
  - Validate that the running services match the startup configuration

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 82

Back when Unix systems had very low processing and memory capacity, the inetd server was used to listen on all required TCP/UDP ports and spawn the associated server executables only when needed. Xinetd is a more modern inetd which includes TCP wrappers firewalling by default (TCP wrappers had to be invoked specifically by inetd using the tcpd executable). Some administrators still install xinetd as a way of using TCP wrappers to limit access to the services on a host. While not exactly common today, inetd and xietd are both legitimate means to start a service on-demand.

The last way of starting a service is not okay, if you care about the availability of the service. Usually it works something like this: the administrator downloads and build the daemon executable from scratch and then performs manual testing of the service by running it from the command shell, reconfiguring and restarting as needed (so far, we're okay). At some point, when she has the daemon configured the way she wants, she just leaves it running and forgets to create the associated service file to start it automatically on system boot. Since Linux systems sometime go for long periods of time between boots, this configuration flaw may not show itself until the system is rebooted and the service is suddenly not running! As an auditor, we will compare the list of running services to those configured to run at startup to ensure that there are no "extra" services.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021

Using Systemd to View Services

- The main command to control Systemd is `systemctl`
  - List configured services:
    `systemctl list-units`
  - List all services:
    `systemctl -a`
- Services are managed using `systemctl`
  - Start a service:
    `systemctl start <servicename>`
  - Enable a service permanently:
    `systemctl enable <servicename>`

Systemd provides a single tool as a consistent interface to the startup system. While there are still configuration files and startup scripts, managing them, starting services, stopping services, enabling a service permanently, and disabling a service are all accomplished through the `systemctl` tool.

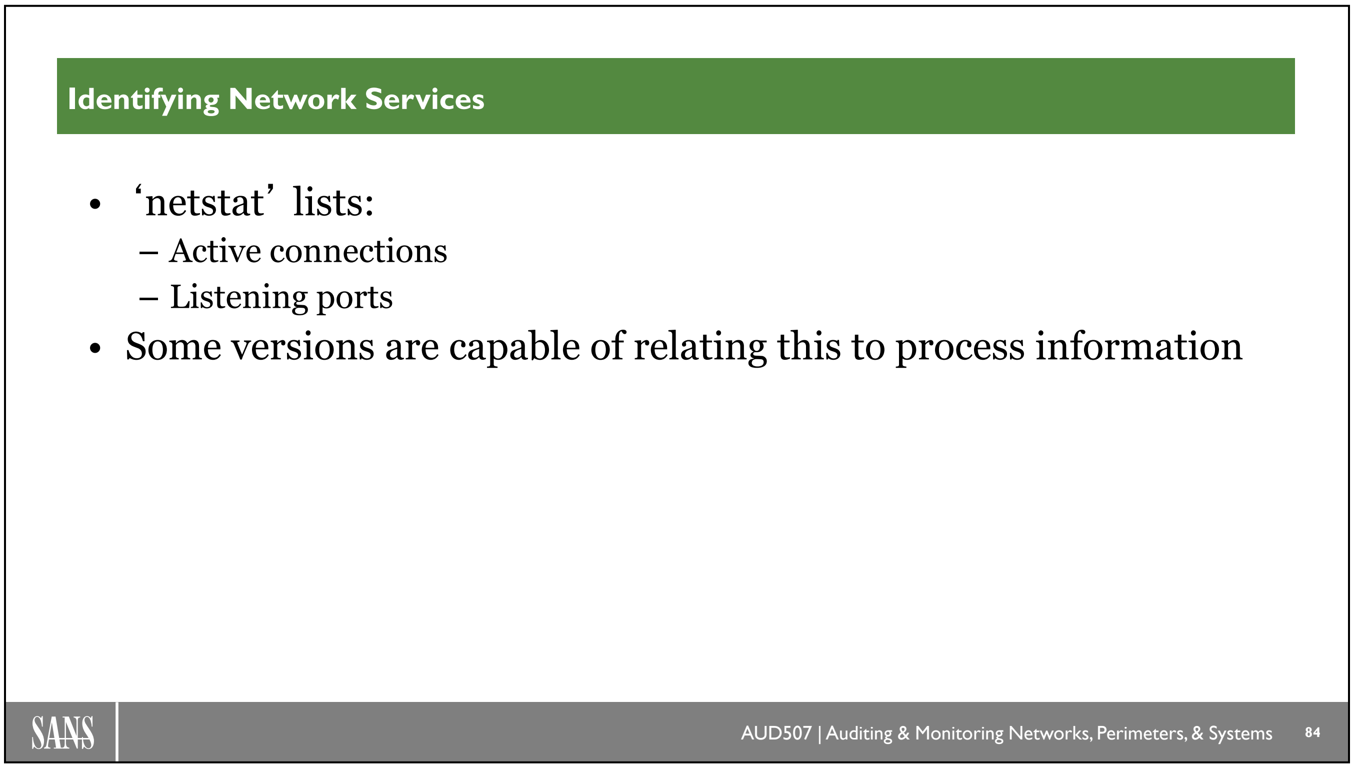
The `list-units` option will show all currently configured services. These are services that will be run or started upon system reboot, even if they are not currently running. If, instead, you would like to obtain a list of all services that are installed, even if they are not running, the –a option will provide this list. This list, along with the configuration status of each service, should be included in the baseline for the system.

Services can be started and stopped manually using the 'start' and 'stop' options. Simply starting a service does not mean that the service will be configured to run automatically. To permanently enable a service, the 'enable' option must be used.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Identifying Network Services

- ‘netstat’ lists:
  - Active connections
  - Listening ports
- Some versions are capable of relating this to process information

SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 84

Netstat lists all active connections in addition to the ports where programs are listening for connections. Simply use the command netstat -a -p --inet for a listing of this information. It is important to note that not all UNIX versions support the ‘-p’ option for netstat, so you might need to use another tool to find the process information. We will provide a solution to that problem shortly. In the example on the next slide, a partial netstat listing is shown.

Note that I've used some special arguments. The -a argument makes netstat display the entries for all sockets, including those that are simply listening. This is useful for identifying all the ports that are open. Using this information, the system administrator can prune system services, eliminating those that aren’t absolutely essential.

August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021

Netstat with Process Information

```
# netstat -antp | grep LISTEN
tcp    0   0 127.0.0.1:25     0.0.0.0:*    LISTEN  2681/sendmail: MTA:
tcp    0   0 0.0.0.0:4443     0.0.0.0:*    LISTEN  2555/vmware-hostd
tcp    0   0 0.0.0.0:443      0.0.0.0:*    LISTEN  1543/nginx: master
tcp    0   0 0.0.0.0:8443     0.0.0.0:*    LISTEN  1543/nginx: master
tcp    0   0 0.0.0.0:8000     0.0.0.0:*    LISTEN  1543/nginx: master
tcp    0   0 127.0.0.1:587    0.0.0.0:*    LISTEN  2681/sendmail: MTA:
tcp    0   0 0.0.0.0:80       0.0.0.0:*    LISTEN  1543/nginx: master
tcp    0   0 127.0.0.1:8307   0.0.0.0:*    LISTEN  2555/vmware-hostd
tcp    0   0 0.0.0.0:22       0.0.0.0:*    LISTEN  1784/sshd
tcp    0   0 127.0.0.1:631    0.0.0.0:*    LISTEN  6856/cupsd
tcp6   0   0 :::4443          :::*         LISTEN  2555/vmware-hostd
tcp6   0   0 :::902           :::*         LISTEN  2548/vmware-authdla
tcp6   0   0 :::8080          :::*         LISTEN  2122/java
tcp6   0   0 :::80            :::*         LISTEN  1543/nginx: master
tcp6   0   0 :::22            :::*         LISTEN  1784/sshd
tcp6   0   0 ::1:631          :::*         LISTEN  6856/cupsd
```

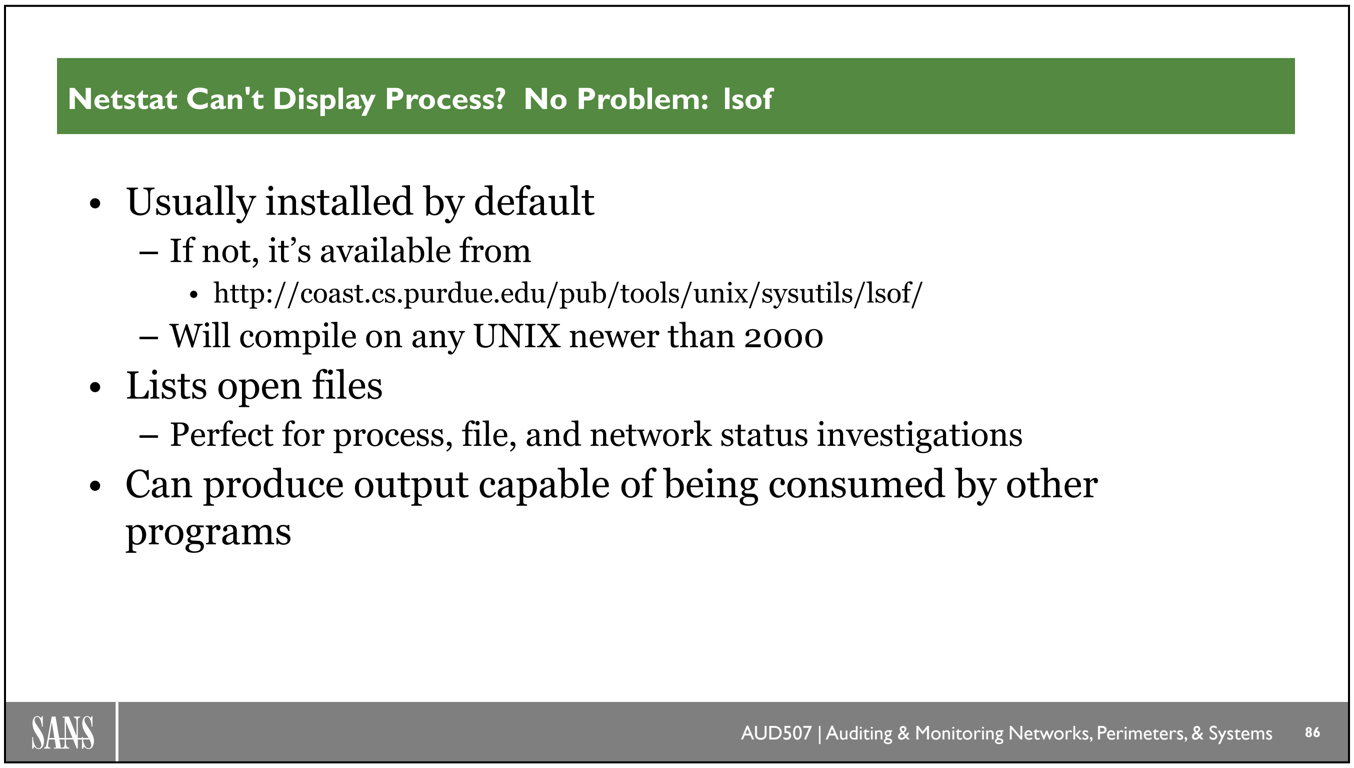
Netstat is a fairly standard tool to find on most operating systems these days. Even though it's known as a UNIX tool, it is actually more a part of the IPv4 and IPv6 implementation of Berkeley sockets. For this reason, you'll not only find it on POSIX-compliant systems, but you will also find it on mainframe operating systems, legacy mini-computers, and more.

The '-p' option is the primary point for this slide. You can see that the process holding open each port is identified for us. Although the '-p' option is extremely valuable, it won't always be available! For example, you will find that OS X and other similar systems lack the option completely! What can be done in a case like this to map network ports to running processes?

August 10, 2021

August 10, 2021

© SANS Institute 2021



Netstat Can't Display Process? No Problem: lsof

- Usually installed by default
  - If not, it's available from
    - http://coast.cs.purdue.edu/pub/tools/unix/sysutils/lsof/
  - Will compile on any UNIX newer than 2000
- Lists open files
  - Perfect for process, file, and network status investigations
- Can produce output capable of being consumed by other programs

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 86

Lsof to the rescue! Because everything in UNIX is a file, including network ports and connections, and because lsof will list open files, this tool can be used to list all files that are network ports and are currently open.

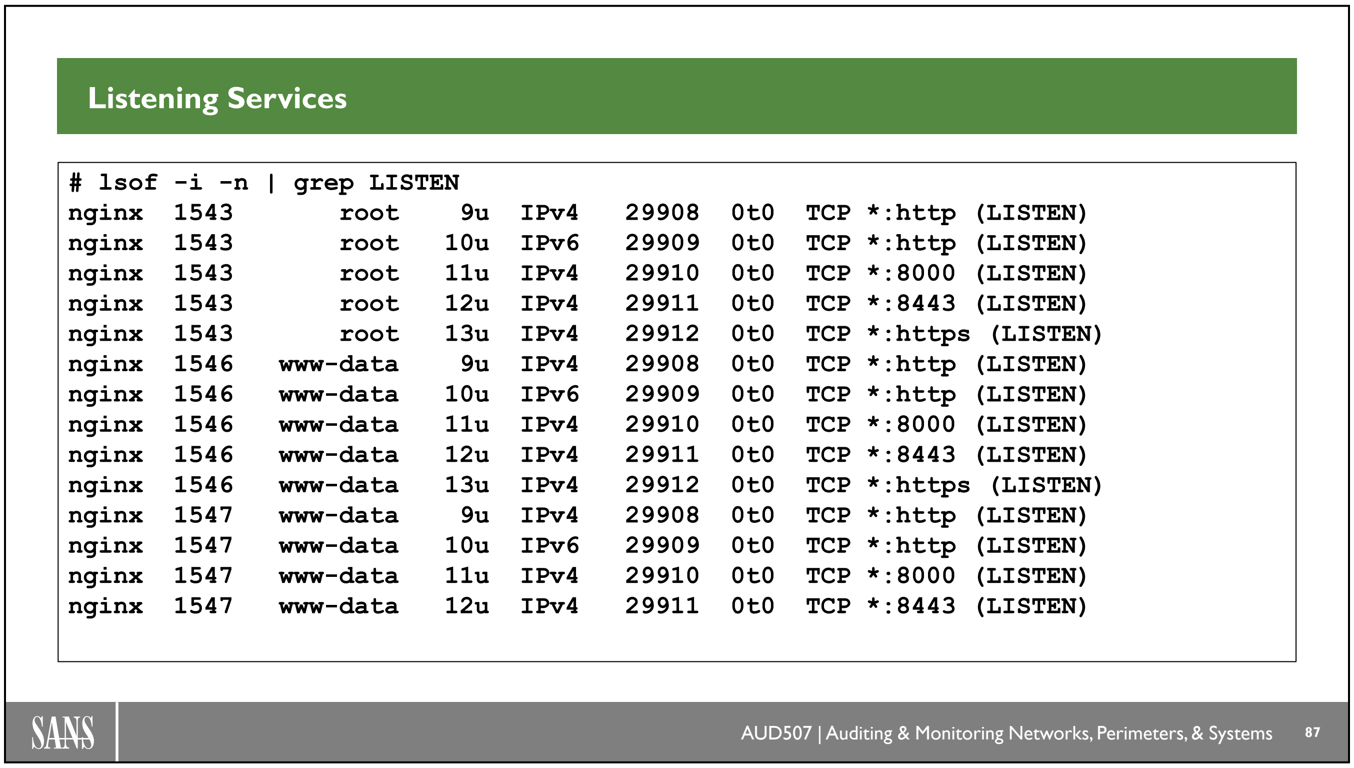
This tool has become a part of the standard base for UNIX these days. Even so, you may find some systems that do not have the lsof tool. If you happen to run across one, you can still obtain the source code for lsof from Purdue University for free. It can then be compiled for the specific platform you're dealing with and added to your audit ISO.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Listening Services

```
# lsof -i -n | grep LISTEN
nginx  1543       root    9u  IPv4   29908  0t0  TCP *:http (LISTEN)
nginx  1543       root   10u  IPv6   29909  0t0  TCP *:http (LISTEN)
nginx  1543       root   11u  IPv4   29910  0t0  TCP *:8000 (LISTEN)
nginx  1543       root   12u  IPv4   29911  0t0  TCP *:8443 (LISTEN)
nginx  1543       root   13u  IPv4   29912  0t0  TCP *:https (LISTEN)
nginx  1546   www-data    9u  IPv4   29908  0t0  TCP *:http (LISTEN)
nginx  1546   www-data   10u  IPv6   29909  0t0  TCP *:http (LISTEN)
nginx  1546   www-data   11u  IPv4   29910  0t0  TCP *:8000 (LISTEN)
nginx  1546   www-data   12u  IPv4   29911  0t0  TCP *:8443 (LISTEN)
nginx  1546   www-data   13u  IPv4   29912  0t0  TCP *:https (LISTEN)
nginx  1547   www-data    9u  IPv4   29908  0t0  TCP *:http (LISTEN)
nginx  1547   www-data   10u  IPv6   29909  0t0  TCP *:http (LISTEN)
nginx  1547   www-data   11u  IPv4   29910  0t0  TCP *:8000 (LISTEN)
nginx  1547   www-data   12u  IPv4   29911  0t0  TCP *:8443 (LISTEN)
```

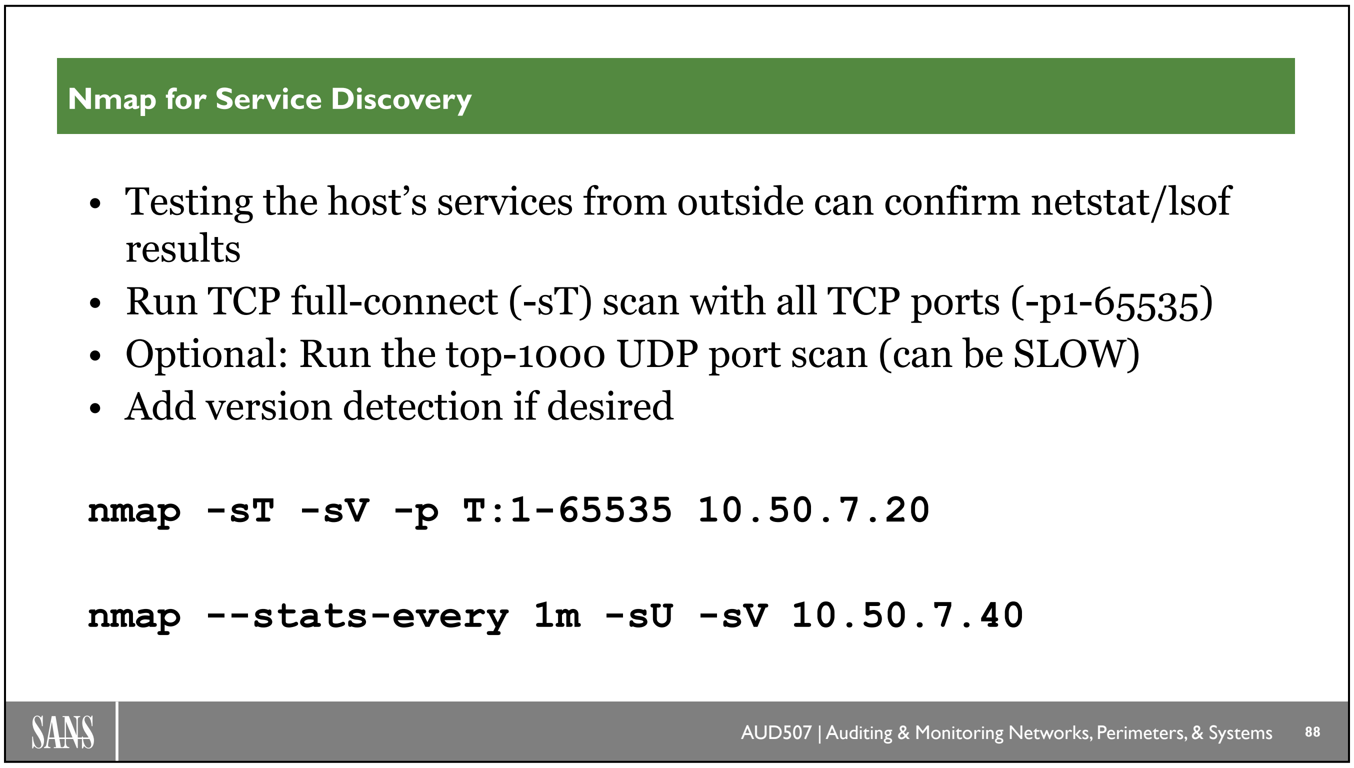
As you can see, lsof -i displays a listing that is similar to that of the netstat command discussed earlier.

Although there are certain columns, like the PID, that will change, much of this information should be a part of a baseline. In fact, you likely have in mind that the administrator should be very interested in using a baseline script to perform periodic automatic analysis of his system in order to have an early warning of unauthorized operational changes.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Nmap for Service Discovery

- Testing the host's services from outside can confirm netstat/lsof results
- Run TCP full-connect (-sT) scan with all TCP ports (-p1-65535)
- Optional: Run the top-1000 UDP port scan (can be SLOW)
- Add version detection if desired

```
nmap -sT -sV -p T:1-65535 10.50.7.20

nmap --stats-every 1m -sU -sV 10.50.7.40
```

Testing the services running from outside the host is a very good idea. It allows you to confirm the results of the local commands, and provides a form of centralized monitoring. TCP scanning should be done with the TCP full connect option on all TCP ports. UDP scanning can be much slower and full scanning may not be feasible in limited scan windows. A top-1000 port scan (Nmap's default) is usually sufficient to identify running services.

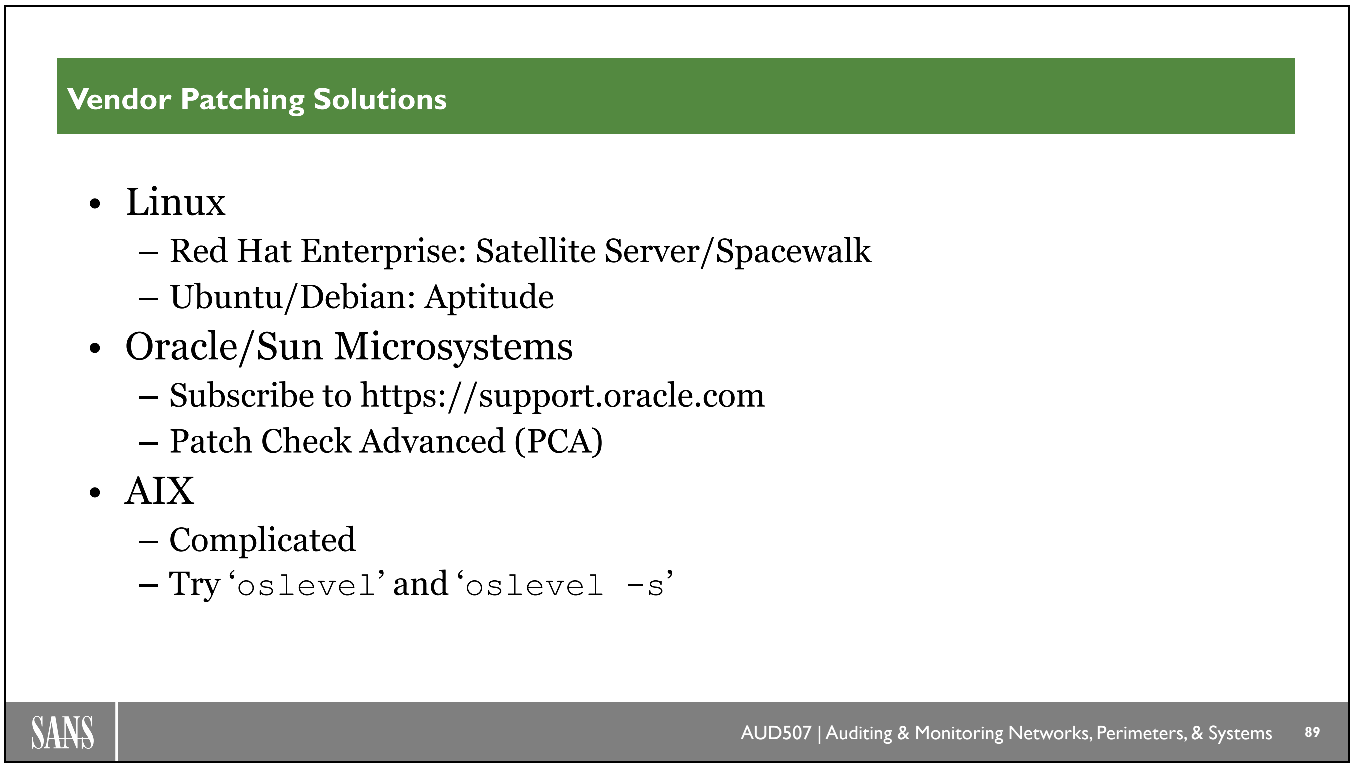
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Vendor Patching Solutions

- Linux
  - Red Hat Enterprise: Satellite Server/Spacewalk
  - Ubuntu/Debian: Aptitude
- Oracle/Sun Microsystems
  - Subscribe to https://support.oracle.com
  - Patch Check Advanced (PCA)
- AIX
  - Complicated
  - Try ‘`oslevel`’ and ‘`oslevel -s`’

The two most common types of UNIX to come across in an enterprise these days are Linux (in many shapes and sizes) and Solaris. Although there is an enormous number of different Linux variants out there, most of them fall into two major groups: Those that are derived from Debian and those that are derived from Red Hat.

If your organization is using Red Hat Enterprise, you might want to look into a Satellite server. This system allows you to push out patches, centrally manage users via Kerberos, and manage general settings from a central console. In terms of patching, this is somewhat manageable as long as all of your software was installed using RPMs via Satellite.

For Debian-derived systems, the primary tool that can easily be used to check on patch status is Aptitude. Unfortunately, there's nothing here to centrally manage what happens, and you do have to rely on the repository that your systems are pointed at to be up to date, but for software installed using the apt toolkit, Aptitude does a nice job.

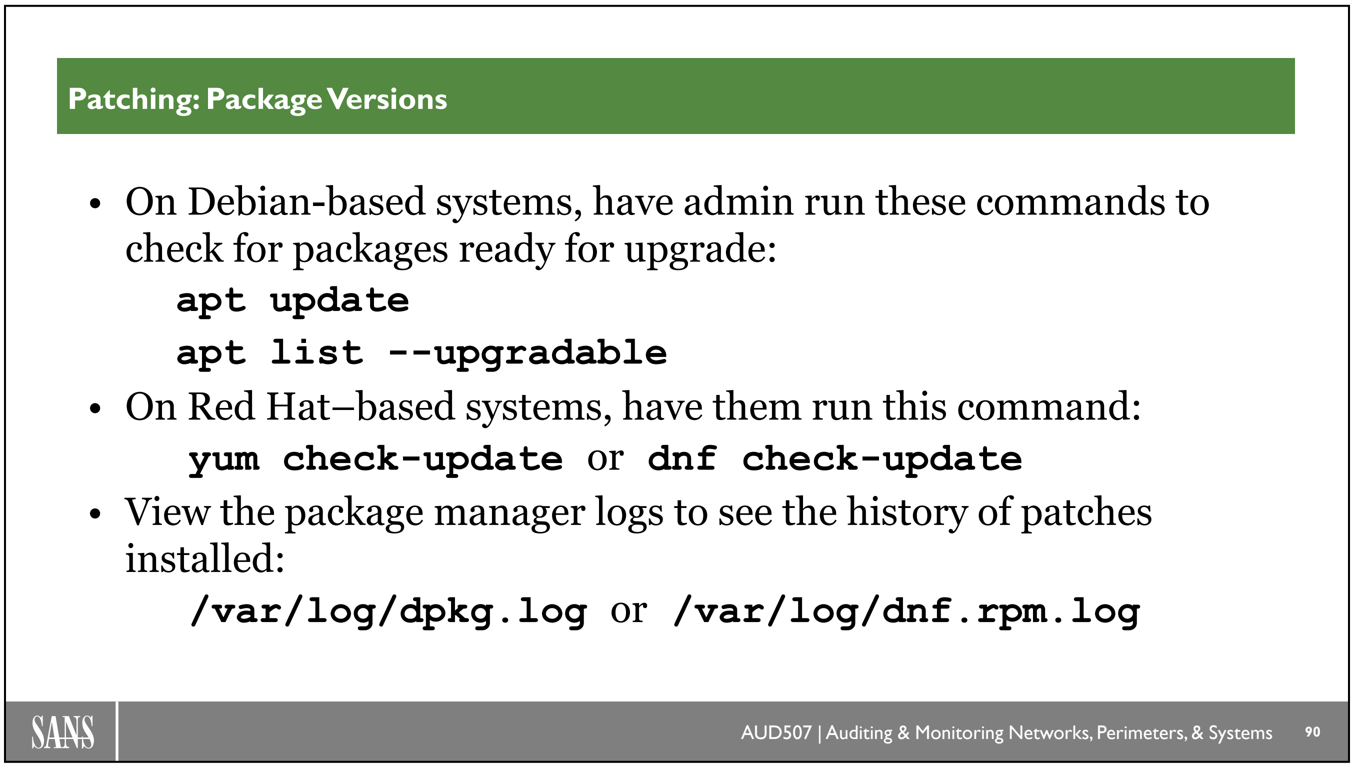
August 10, 2021

In the Sun/Oracle world, the first thing that should happen is that an administrator should subscribe to the SunSolve alerts system. A handy tool called “Patch Check Advanced,” written by Martin Paul, can help. This free tool for your modern Solaris systems can be obtained from http://www.par.univie.ac.at/solaris/pca/.

AIX is one of the more difficult systems for patch management. The oslevel command will give you information about the “Maintenance Level” of the system. Adding the ‘-s’ option will give you service pack information as well. Get more information at:
https://www.ibm.com/support/knowledgecenter/en/ssw_aix_72/com.ibm.aix.cmds4/oslevel.htm

August 10, 2021

© SANS Institute 2021



Patching: Package Versions

- On Debian-based systems, have admin run these commands to check for packages ready for upgrade:

  ```
  apt update
  apt list --upgradable
  ```

- On Red Hat–based systems, have them run this command:

  `yum check-update` or **`dnf check-update`**

- View the package manager logs to see the history of patches installed:

  `/var/log/dpkg.log` or **`/var/log/dnf.rpm.log`**

SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 90

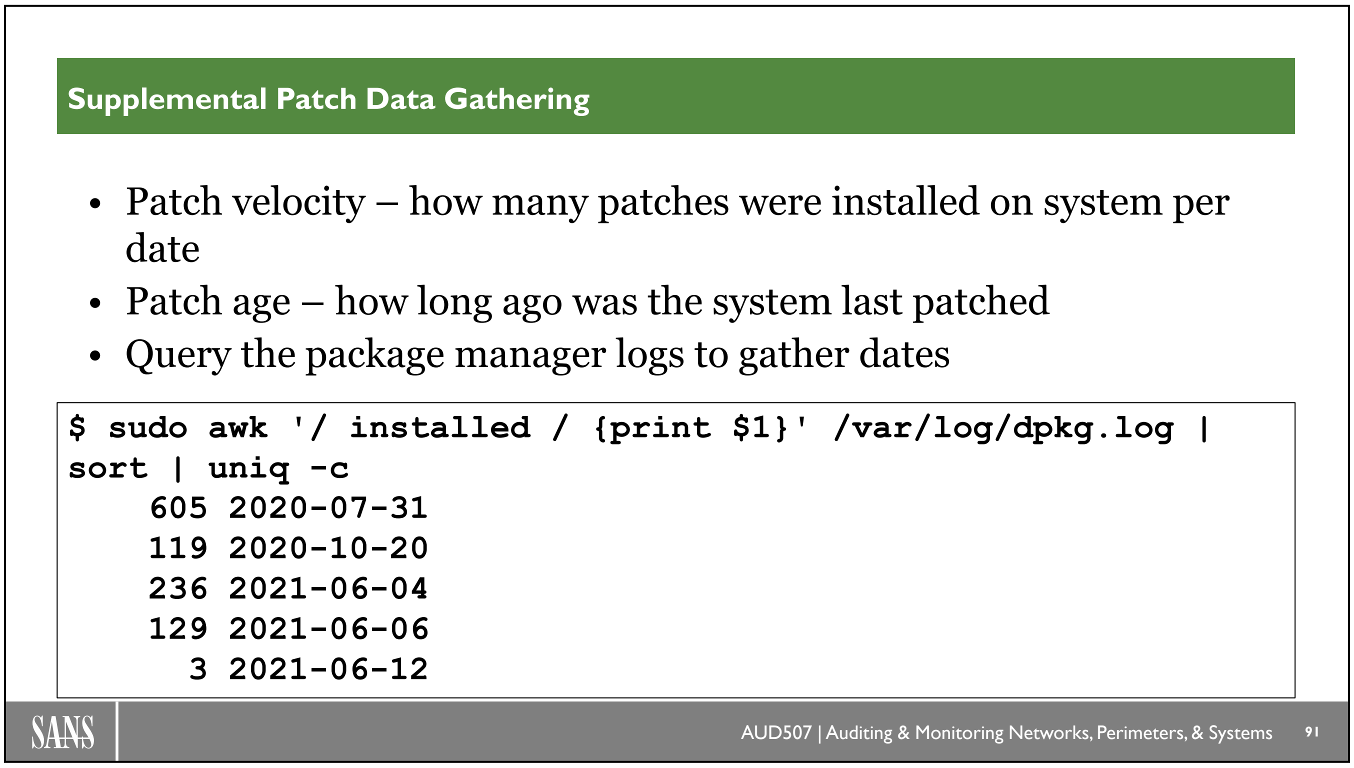
For software that was installed by a package manager on Linux, it's pretty easy to check which packages have an upgrade available. The major package management tools can update their local database of package data from the central repositories, and then report on any installed packages which are out of date.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Supplemental Patch Data Gathering

- Patch velocity – how many patches were installed on system per date
- Patch age – how long ago was the system last patched
- Query the package manager logs to gather dates

```
$ sudo awk '/ installed / {print $1}' /var/log/dpkg.log |
sort | uniq -c
    605 2020-07-31
    119 2020-10-20
    236 2021-06-04
    129 2021-06-06
      3 2021-06-12
```

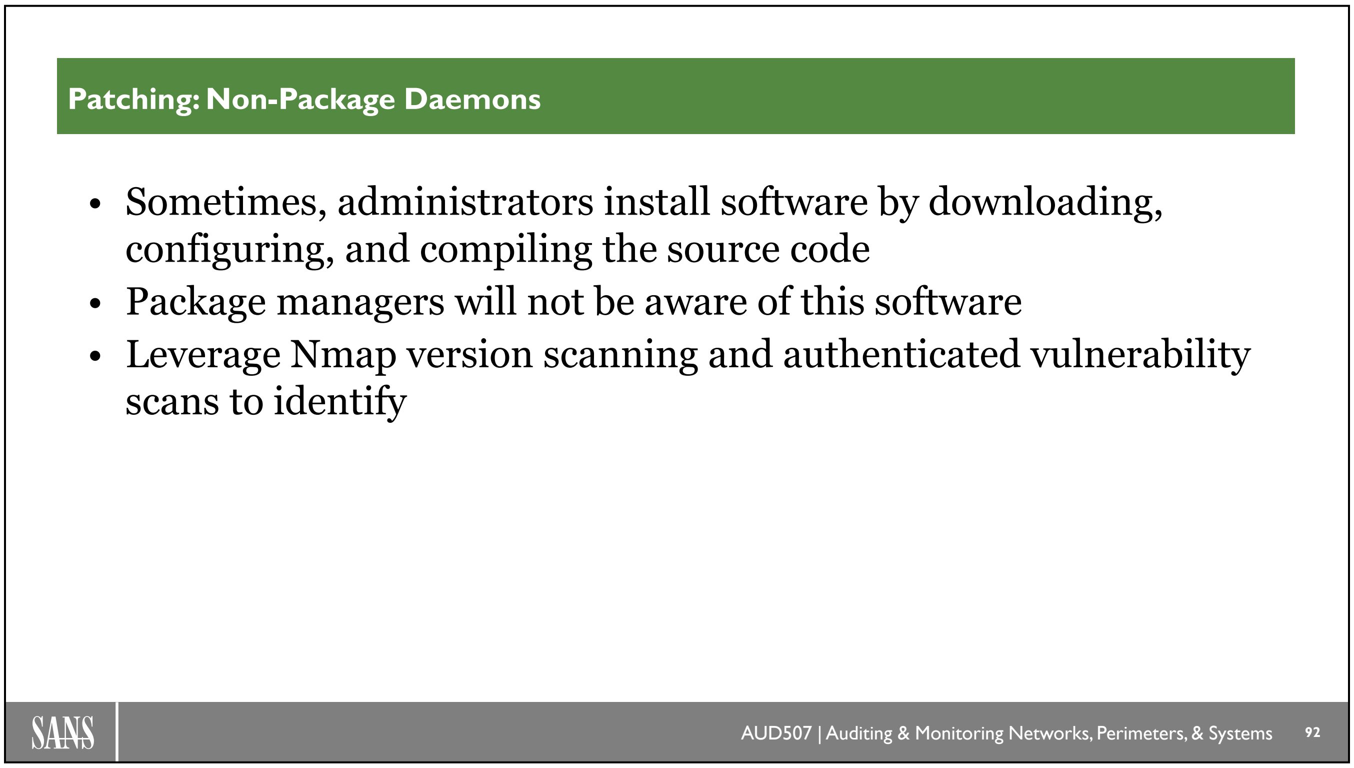
In out audits, we often use two quick measurements to gauge the patching history of a system. "Patch velocity," measured on an Ubuntu server in this slide, is a count of patches installed per date on the system. A quick glance at these results shows us that the server is not being regularly patched.

"Patch age" is the number of days it has been since the host was last patched. If the patch age is quite high, it is likely that the host is missing patches. Neither of these tests are as accurate or conclusive as the results returned by a patching tool or authenticated vulnerability scan, but they can serve to reduce the uncertainty surround patching on the host.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Patching: Non-Package Daemons

- Sometimes, administrators install software by downloading, configuring, and compiling the source code
- Package managers will not be aware of this software
- Leverage Nmap version scanning and authenticated vulnerability scans to identify

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 92

Very often, the version of an important piece of software which is available from a distribution's package repositories is not appropriate for an administrator's needs. In these cases, they will often download and install the software themselves from source code.

When this is done, the package manager will not be aware that the software is installed and would not be able to upgrade it, anyway. In these cases, the most effective way of monitoring for outdated services is to use the version detection scripts on Nmap, or even better, the authenticated scanning feature of a vulnerability scanner.

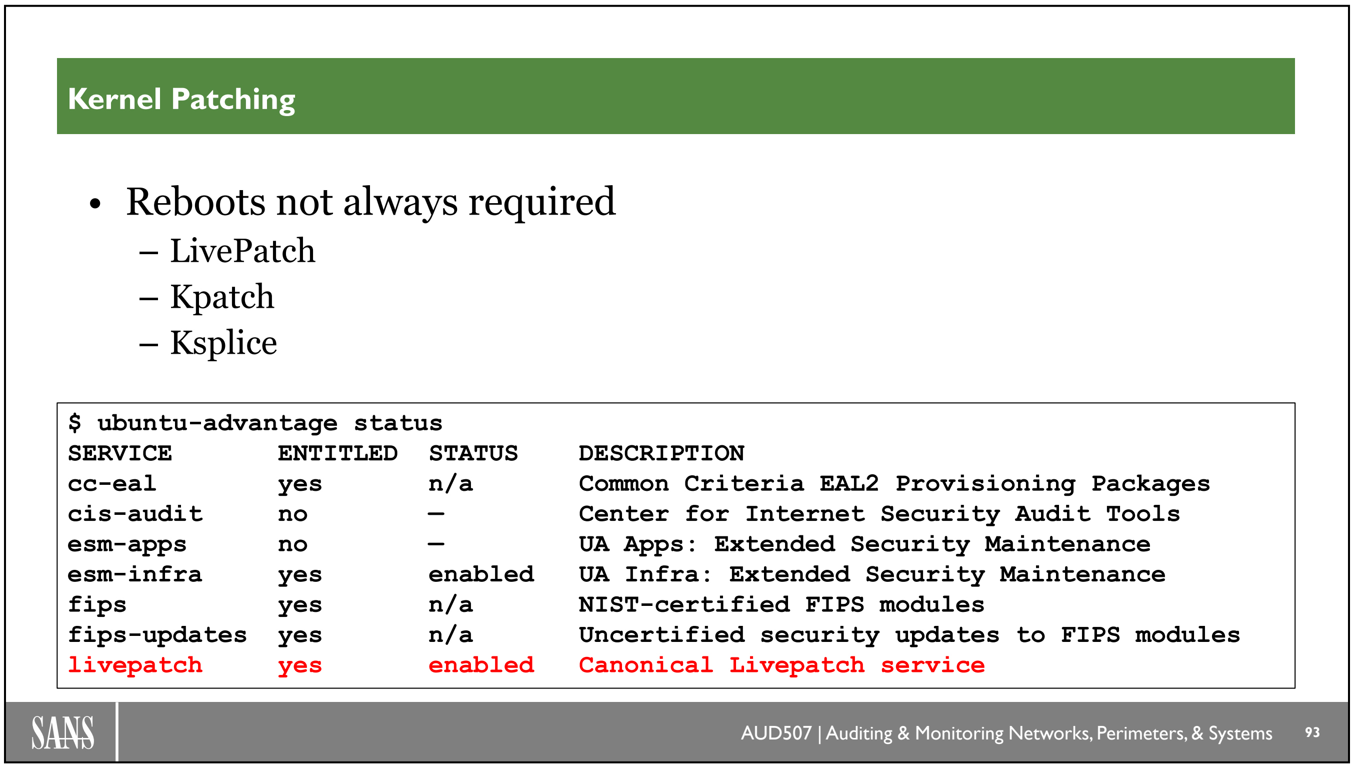
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Kernel Patching

- Reboots not always required
  - LivePatch
  - Kpatch
  - Ksplice

```
$ ubuntu-advantage status
SERVICE       ENTITLED  STATUS    DESCRIPTION
cc-eal        yes       n/a       Common Criteria EAL2 Provisioning Packages
cis-audit     no        —         Center for Internet Security Audit Tools
esm-apps      no        —         UA Apps: Extended Security Maintenance
esm-infra     yes       enabled   UA Infra: Extended Security Maintenance
fips          yes       n/a       NIST-certified FIPS modules
fips-updates  yes       n/a       Uncertified security updates to FIPS modules
livepatch     yes       enabled   Canonical Livepatch service
```

SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 93

Newer Linux distributions allow for kernel patching without rebooting the server. The new kernel is installed and loaded into memory, and then one at a time, running processes are migrated to using system calls in the new kernel. When the old kernel is no longer being used by any running service, it is shut down.

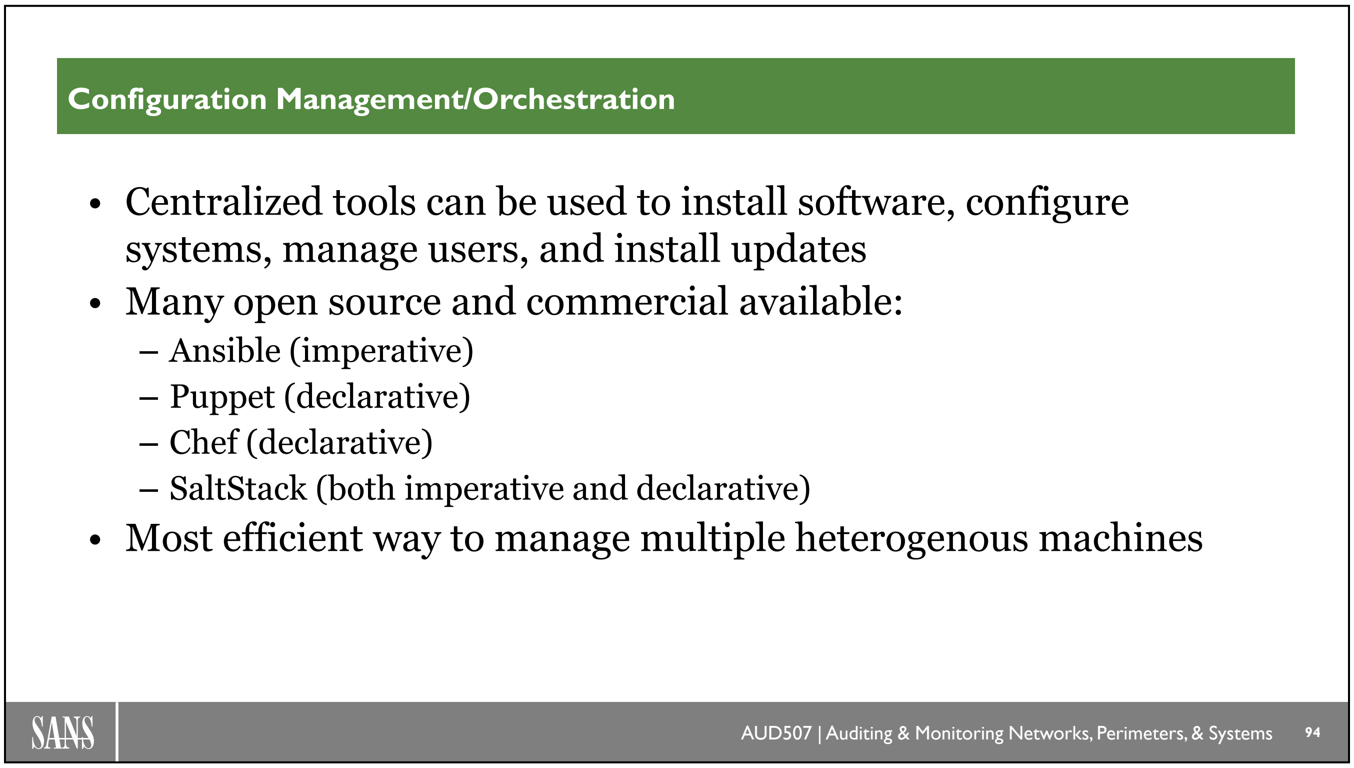
These systems, a few of which are mentioned on the slide, make it easier to install patches on critical servers which cannot easily be shut down for maintenance.

Validation of these services can usually be done at the command line. In the screenshot above, we are checking on the status of LivePatch on an Ubuntu server system using the Ubuntu Advantage command-line tool.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Configuration Management/Orchestration

- Centralized tools can be used to install software, configure systems, manage users, and install updates
- Many open source and commercial available:
  - Ansible (imperative)
  - Puppet (declarative)
  - Chef (declarative)
  - SaltStack (both imperative and declarative)
- Most efficient way to manage multiple heterogenous machines

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 94

Manual management of configurations and installed software on multiple servers running multiple Linux distributions is fraught with peril. Orchestration systems, like Chef and Puppet, or SaltStack, can be a lifesaver for administrators.

These tools provide centralized control of the most important aspects of system configuration management. They can also make it much faster to deploy known-secure systems by performing the baseline configuration on new systems automatically.

Declarative orchestration systems use agents running on the host to ensure that the host's state remains consistent with its roles. This minimizes the work an admin needs to do, but it has the downside of requiring an agent to be running on each managed machine.

Imperative tools connect to the host from a server (usually via SSH or Windows remote management - WinRM), and run the commands required to bring the host into its desired state. An Ansible "playbook" would be an example of this. SaltStack uses a combination of techniques: the requisites system is declarative, but the ordered execution feature is imperative.

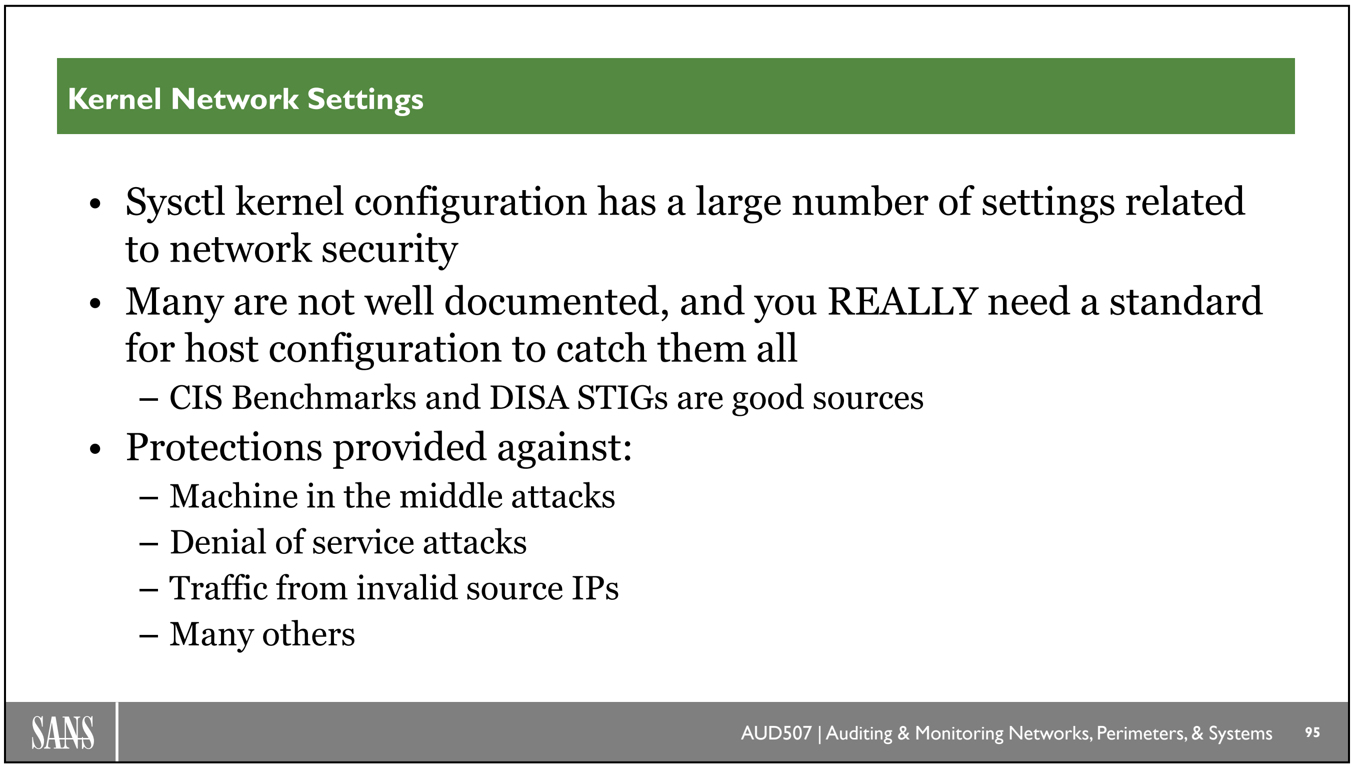
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Kernel Network Settings

- Sysctl kernel configuration has a large number of settings related to network security
- Many are not well documented, and you REALLY need a standard for host configuration to catch them all
  - CIS Benchmarks and DISA STIGs are good sources
- Protections provided against:
  - Machine in the middle attacks
  - Denial of service attacks
  - Traffic from invalid source IPs
  - Many others

The sysctl system which is used to configure the behavior of the running kernel has a LOT of settings related to protecting our Linux hosts against network-based attacks. We have found that the most comprehensive recommendations for these settings comes from consensus-based standards like the CIS benchmarks for Linux or the DISA STIG documents.

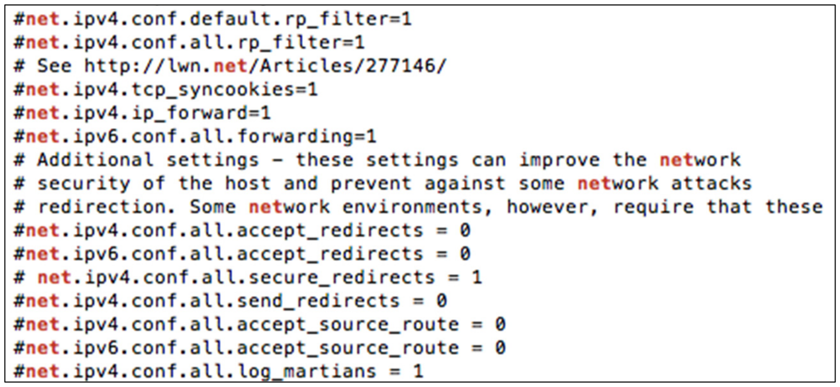
It's fortunate that such resources exist! As you will see on the next few slides, many of these settings are not well documented in the man pages, so it's possible that even well-intentioned administrators may not know what should be configured or how.

August 10, 2021

© SANS Institute 2021

Linux as the Example

- Reconfiguration possible on all, but we have Linux handy
  - /etc/sysctl.conf



```
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1
# See http://lwn.net/Articles/277146/
#net.ipv4.tcp_syncookies=1
#net.ipv4.ip_forward=1
#net.ipv6.conf.all.forwarding=1
# Additional settings - these settings can improve the network
# security of the host and prevent against some network attacks
# redirection. Some network environments, however, require that these
#net.ipv4.conf.all.accept_redirects = 0
#net.ipv6.conf.all.accept_redirects = 0
# net.ipv4.conf.all.secure_redirects = 1
#net.ipv4.conf.all.send_redirects = 0
#net.ipv4.conf.all.accept_source_route = 0
#net.ipv6.conf.all.accept_source_route = 0
#net.ipv4.conf.all.log_martians = 1
```

All the major UNIX systems can be reconfigured in this regard, but how to do it or how to check it will vary by system. Let's look at Linux because it's so handy.

Many of the items we are interested in are controlled through the sysctl (System Control) configuration. This file is typically found in /etc/sysctl.conf. You might not find all the options listed in this slide in this file, but they can be put there if you are using a modern version of Linux. Take a look and see how many of these you can identify. Your instructor will take some time to explain some of them as well.

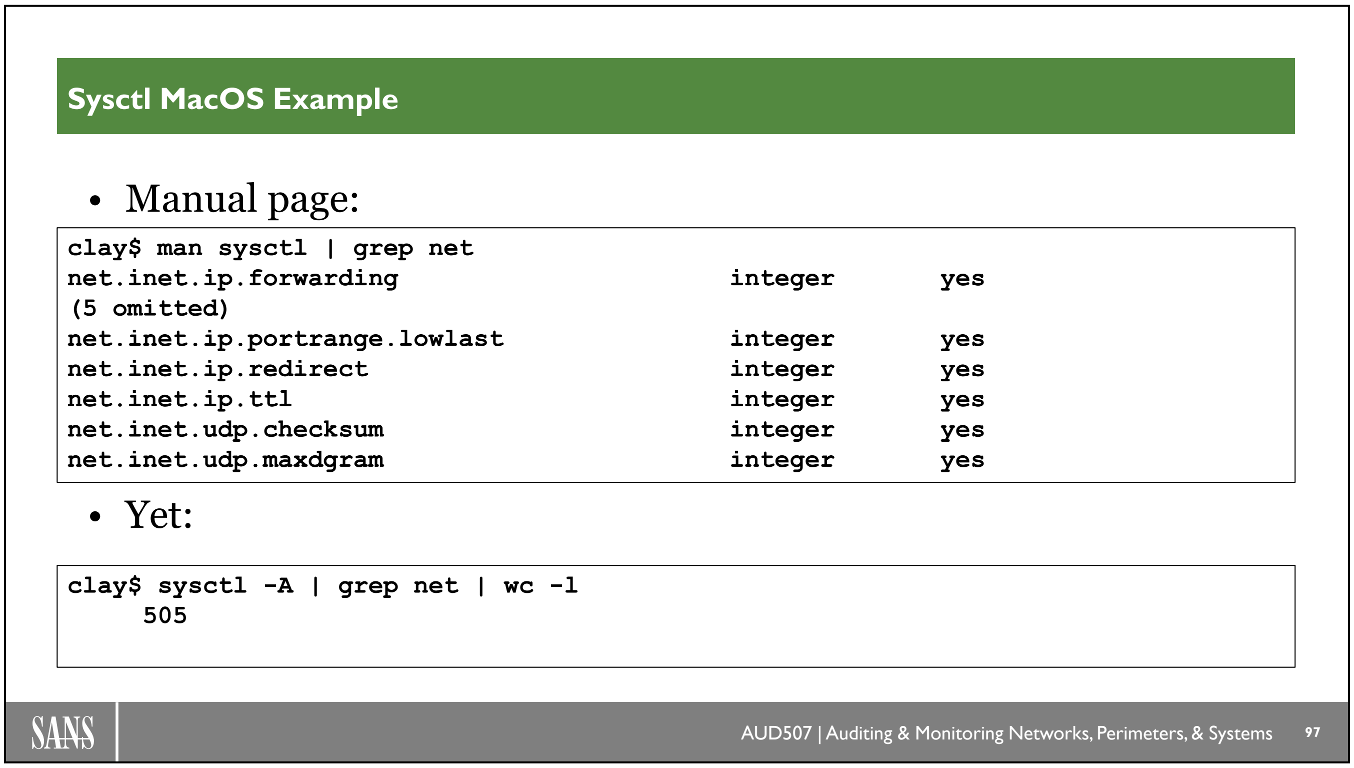
Could someone, such as a security analyst or a system administrator, determine which settings would be appropriate based on our organizational standards for network security? Could we script a test for these settings?

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Sysctl MacOS Example

- Manual page:

```
clay$ man sysctl | grep net
net.inet.ip.forwarding                      integer       yes
(5 omitted)
net.inet.ip.portrange.lowlast               integer       yes
net.inet.ip.redirect                        integer       yes
net.inet.ip.ttl                             integer       yes
net.inet.udp.checksum                       integer       yes
net.inet.udp.maxdgram                       integer       yes
```

- Yet:

```
clay$ sysctl -A | grep net | wc -l
     505
```

As mentioned, what can be controlled with sysctl really does depend on the particular UNIX system. Although there are actually a large number of settings available, the 'man sysctl' manual page lists a significantly lower number. You can see this by comparing the manual page contents from OS X (in the slide) with the output of the 'sysctl –A' command on the same system. What's the story?

Well, as it turns out, manual pages are not always complete. I know that this might not surprise you. At the bottom of the slide, we are running the 'sysctl –A' command (which, by the way, is another great thing to baseline) to inspect the current configuration of all the system parameters. Pulling out network-related items and using "word count" to count them up, we find that there are 505 items available, far more than we found in the manual page!

What's the lesson? Explore… and know your system! This goes for both us as auditors and for the administrators as well. It would be a rare find to have an administrator who is familiar with all 505 settings already - and those are just the network settings!!

August 10, 2021

August 10, 2021

© SANS Institute 2021

Examples of What to Check

- IP Forwarding
- Source routing
- ICMP redirects/IPv6 Router advertisements
- Ignore broadcast ICMP
- Syn Cookies
- **Whatever your enterprise standards require!**

```
$ cat /proc/sys/net/ipv6/conf/all/accept_ra
1
$ sysctl net.ipv6.conf.all.accept_ra
net.ipv6.conf.all.accept_ra = 1
```

There are many kernel settings which contribute to the network security of the host. While we've listed a few on this slide, you should really test your systems against whatever your enterprise standards require.

- If the host is not being used as a router, then IP forwarding should be disabled. Test for a zero value in **`net.ipv6.conf.all.accept_ra`**.
- Source routing is sometimes used by attackers to evade network defenses. The host should be configured to reject source routed packets. Test for a zero value in these settings: **`net.ipv4.conf.all.accept_source_route`**, **`net.ipv4.conf.default.accept_source_route`**, **`net.ipv6.conf.all.accept_source_route`** and **`net.ipv6.conf.default.accept_source_route`**.
- Accepting ICMP redirects or IPv6 router advertisements can make a host vulnerable to machine-in-the-middle attacks. Check these settings for zero values: **`net.ipv4.conf.all.accept_redirects`**, **`net.ipv6.conf.all.accept_redirects`**, **`net.ipv6.conf.all.accept_ra`**, **`net.ipv6.conf.default.accept_ra`**
- The oldie-but-goodie SMURF attack relies on hosts responding to ICMP echo requests sent to the IP broadcast address for the host's subnet. Avoid participating in these attacks by ensuring a value of one in: **`net.ipv4.icmp_echo_ignore_broadcasts`**.
- TCP SYN cookies are used to manage the handling of half-open TCP connections on the host. Enabling this setting helps to prevent denial of service attacks against the TCP stack. Check for a value of one in **`net.ipv4.tcp_syncookies`**.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021

TCP Wrappers

- Installed as a library on most modern Linuxes
  - Check service binary to see if it is linked against **`libwrap.so`**
- Applied by default to xinetd Services
- Are the access controls appropriate

```
$ ldd /usr/sbin/sshd | grep libwrap
...
libwrap.so.0 => /lib/x86_64-linux-gnu/libwrap.so.0
(0x00007f3f129c4000)
```

The tool known as "TCP Wrappers" was developed years ago by Wietse Venema, a well-known UNIX and internet pioneer. This tool acted as a sort of shim between inetd and the service to which users are attempting to connect. The benefit of it was that it allowed us to create network-based access controls to limit who can connect to what on our systems. Today, TCP wrappers is used by default for any service started by Xinetd. It can be linked against most modern daemon executables, also. To test if a binary is linked against libwrap.so, simply use the ldd (list dynamic dependencies) command, as illustrated in the screenshot above.

If TCP wrappers is being used to protect a service, then the host will consider the access controls in the /etc/hosts.deny and hosts.allow files, in addition to any host-based IP firewalls which might be enabled.

August 10, 2021

August 10, 2021

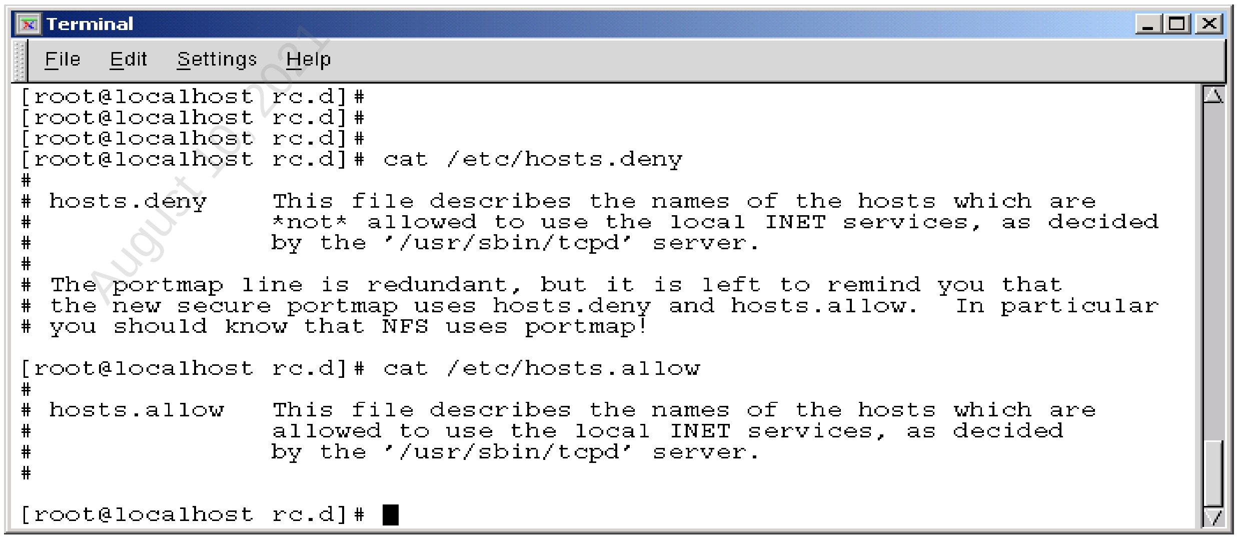
© SANS Institute 2021

TCP Wrappers Access Control

- What do the ACLs say
  - /etc/hosts.deny should include ALL:ALL.
  - /etc/hosts.allow should list individual (trusted) hosts on a service-by-service basis
- By default, TCP Wrappers is installed, but the hosts.deny file is empty
  - Rendered useless

By default, TCP Wrappers comes installed, but it's toothless. To be effective, we must adopt a "deny everything, and then allow back by exception" ACL policy. We recommend that you ensure that you have both a hosts.deny file and a hosts.allow file. The hosts.deny file should be denying all connections to all services. This will implement a default deny policy.

If there is a service that we actually intend to run, it can simply be added to the hosts.allow file. Even if we want everyone in the world to connect to this service, we should still use the hosts.deny and hosts.allow files as described. The reason is that if some other service is inadvertently installed or enabled, perhaps during a patching or other installation process, we want to make sure that the extra service is well defended. Using this method will ensure that someone must actively permit access to the service, or access will be denied.



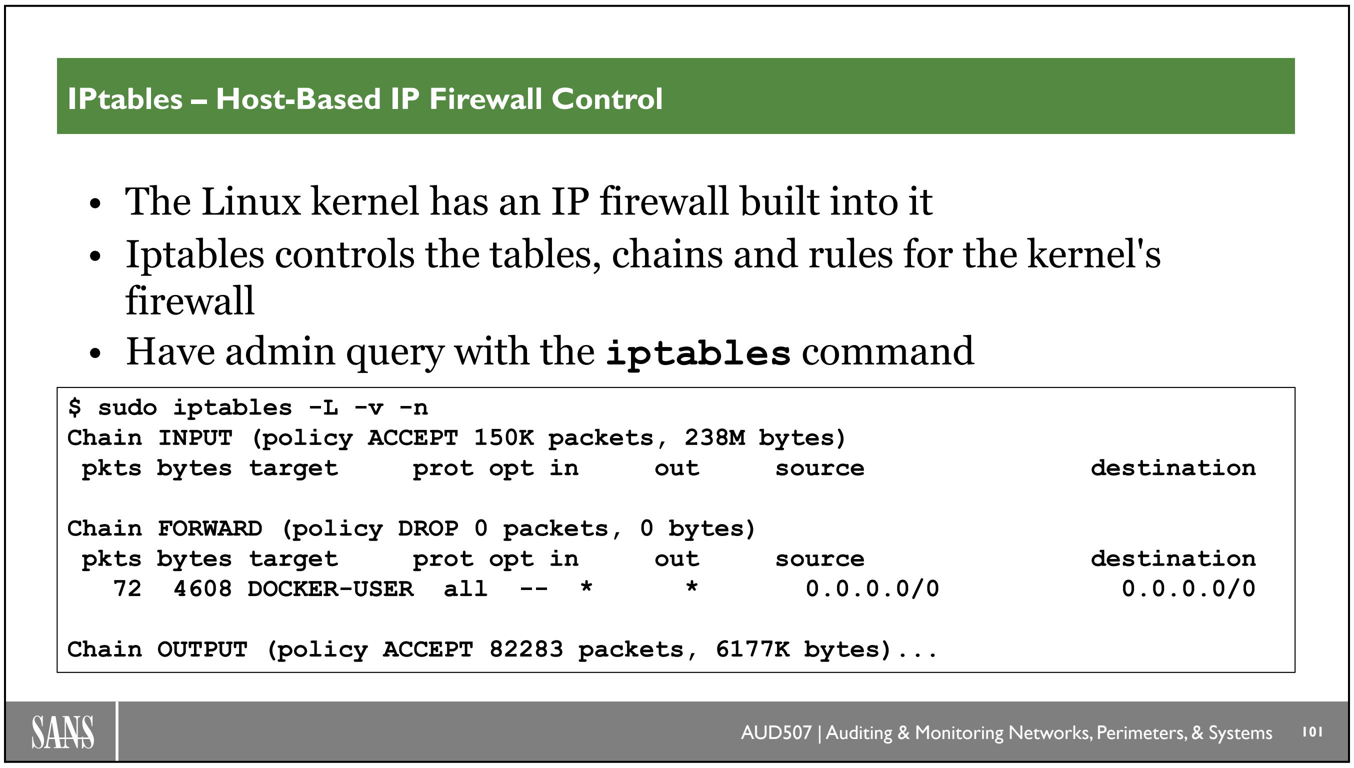
```
[root@localhost rc.d]#
[root@localhost rc.d]#
[root@localhost rc.d]#
[root@localhost rc.d]# cat /etc/hosts.deny
#
# hosts.deny    This file describes the names of the hosts which are
#               *not* allowed to use the local INET services, as decided
#               by the '/usr/sbin/tcpd' server.
#
# The portmap line is redundant, but it is left to remind you that
# the new secure portmap uses hosts.deny and hosts.allow.  In particular
# you should know that NFS uses portmap!

[root@localhost rc.d]# cat /etc/hosts.allow
#
# hosts.allow   This file describes the names of the hosts which are
#               allowed to use the local INET services, as decided
#               by the '/usr/sbin/tcpd' server.
#

[root@localhost rc.d]# █
```

August 10, 2021
Technet24

© SANS Institute 2021



IPtables – Host-Based IP Firewall Control

- The Linux kernel has an IP firewall built into it
- Iptables controls the tables, chains and rules for the kernel's firewall
- Have admin query with the **`iptables`** command

```
$ sudo iptables -L -v -n
Chain INPUT (policy ACCEPT 150K packets, 238M bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   72  4608 DOCKER-USER  all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT 82283 packets, 6177K bytes)...
```

SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 101

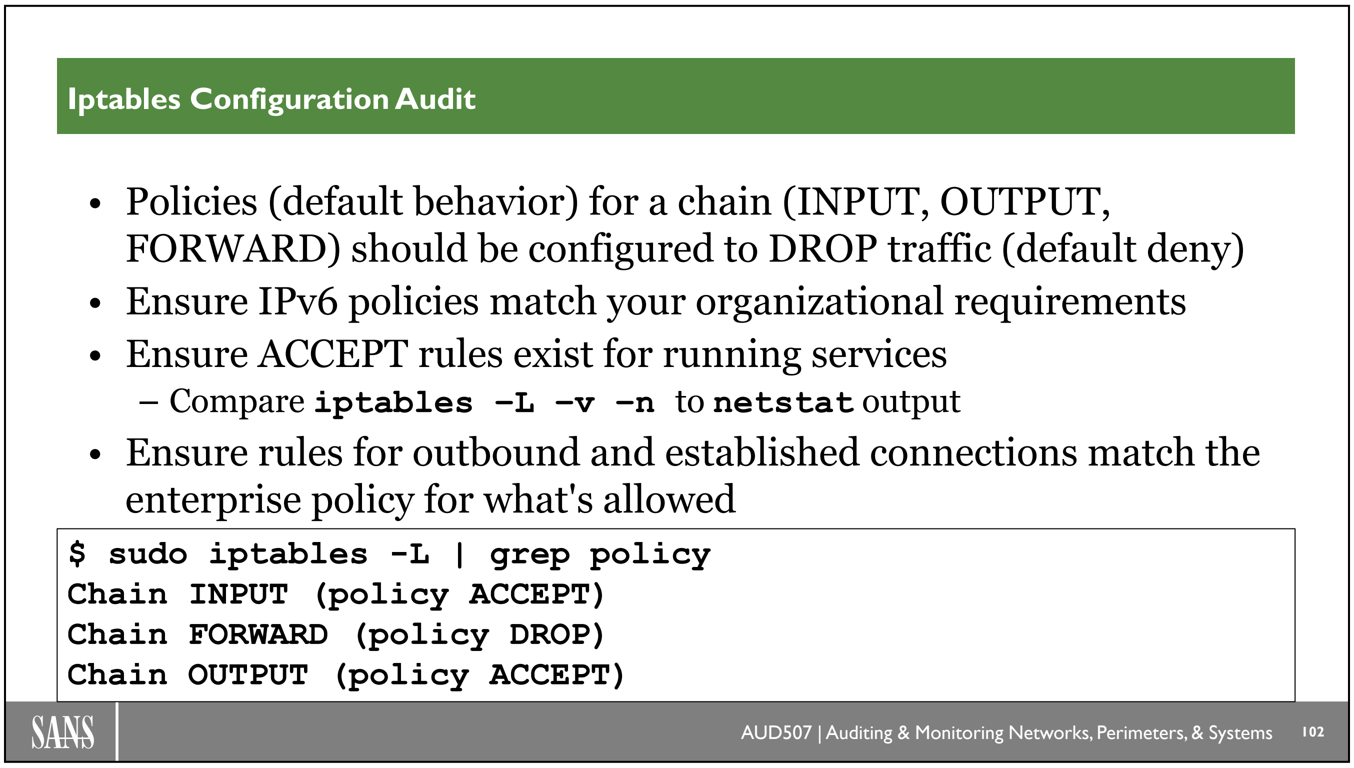
Linux kernels have a built-in IP firewall which is quite capable. It is configured using a tool called **`iptables`**, which an administrator will use to define policies (default behaviors) and specific rules for "chains" like the default INPUT, OUTPUT and FORWARD chains.

The administrator can query the current iptables settings with the **`iptables`** command. Because the settings affect the kernel, root access is required to interact with the **`iptables`** tool.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Iptables Configuration Audit

- Policies (default behavior) for a chain (INPUT, OUTPUT, FORWARD) should be configured to DROP traffic (default deny)
- Ensure IPv6 policies match your organizational requirements
- Ensure ACCEPT rules exist for running services
  - Compare **`iptables -L -v -n`** to **`netstat`** output
- Ensure rules for outbound and established connections match the enterprise policy for what's allowed

```
$ sudo iptables -L | grep policy
Chain INPUT (policy ACCEPT)
Chain FORWARD (policy DROP)
Chain OUTPUT (policy ACCEPT)
```

SANS

We recommend that firewalls should be configured in a default-deny stance. For iptables, this means setting the policy for your host's various chains to DROP (silently discard the packet) or REJECT (send a TCP Reset to the originator of the packet before discarding).

As a minimum firewall audit, we will check that the policies are set to deny all traffic, with allow rules only for permitted services. If a network service is running on the host, there should be an associated permit (ACCEPT) rule for the service. Rules for any outbound connections from the host should match the enterprise policy for allowing outbound traffic.

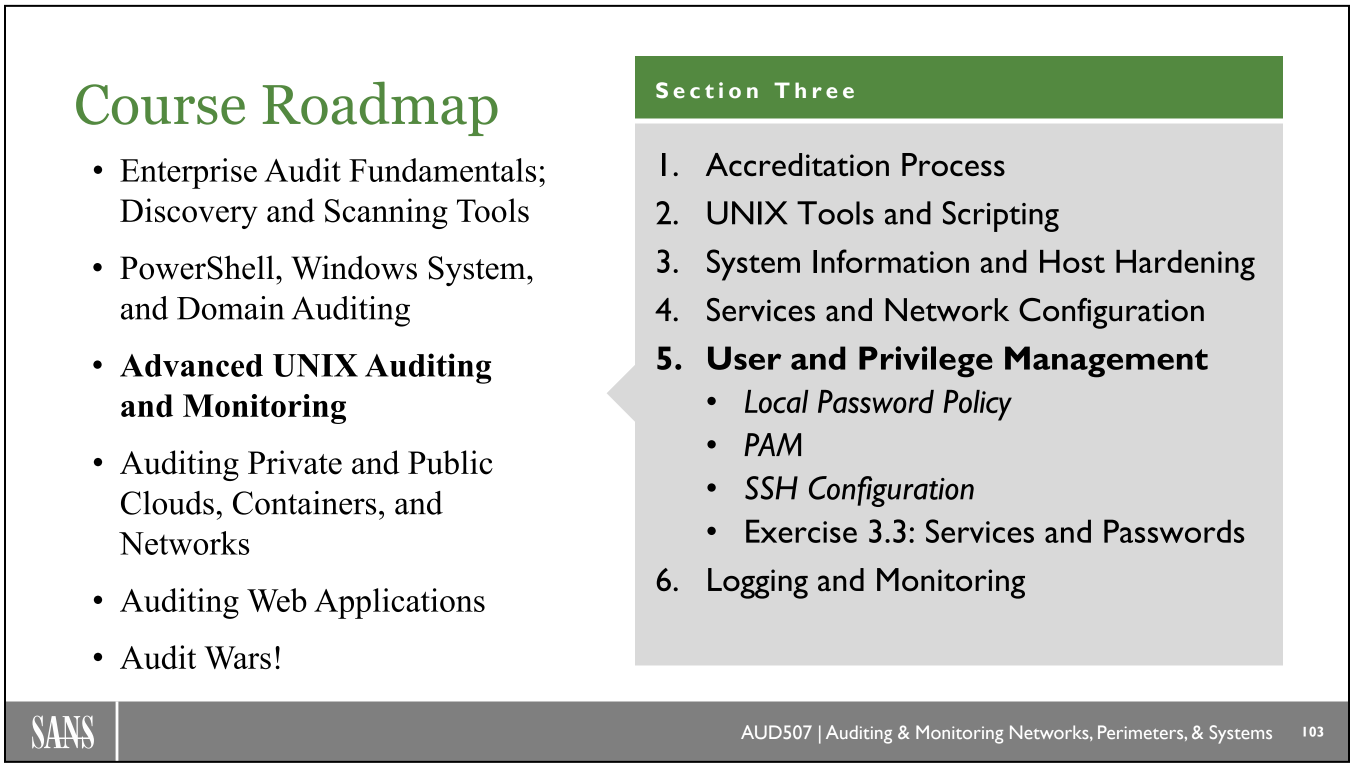
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Course Roadmap

- Enterprise Audit Fundamentals; Discovery and Scanning Tools
- PowerShell, Windows System, and Domain Auditing
- **Advanced UNIX Auditing and Monitoring**
- Auditing Private and Public Clouds, Containers, and Networks
- Auditing Web Applications
- Audit Wars!

Section Three

1. Accreditation Process
2. UNIX Tools and Scripting
3. System Information and Host Hardening
4. Services and Network Configuration
5. **User and Privilege Management**
   - *Local Password Policy*
   - *PAM*
   - *SSH Configuration*
   - Exercise 3.3: Services and Passwords
6. Logging and Monitoring

SANS

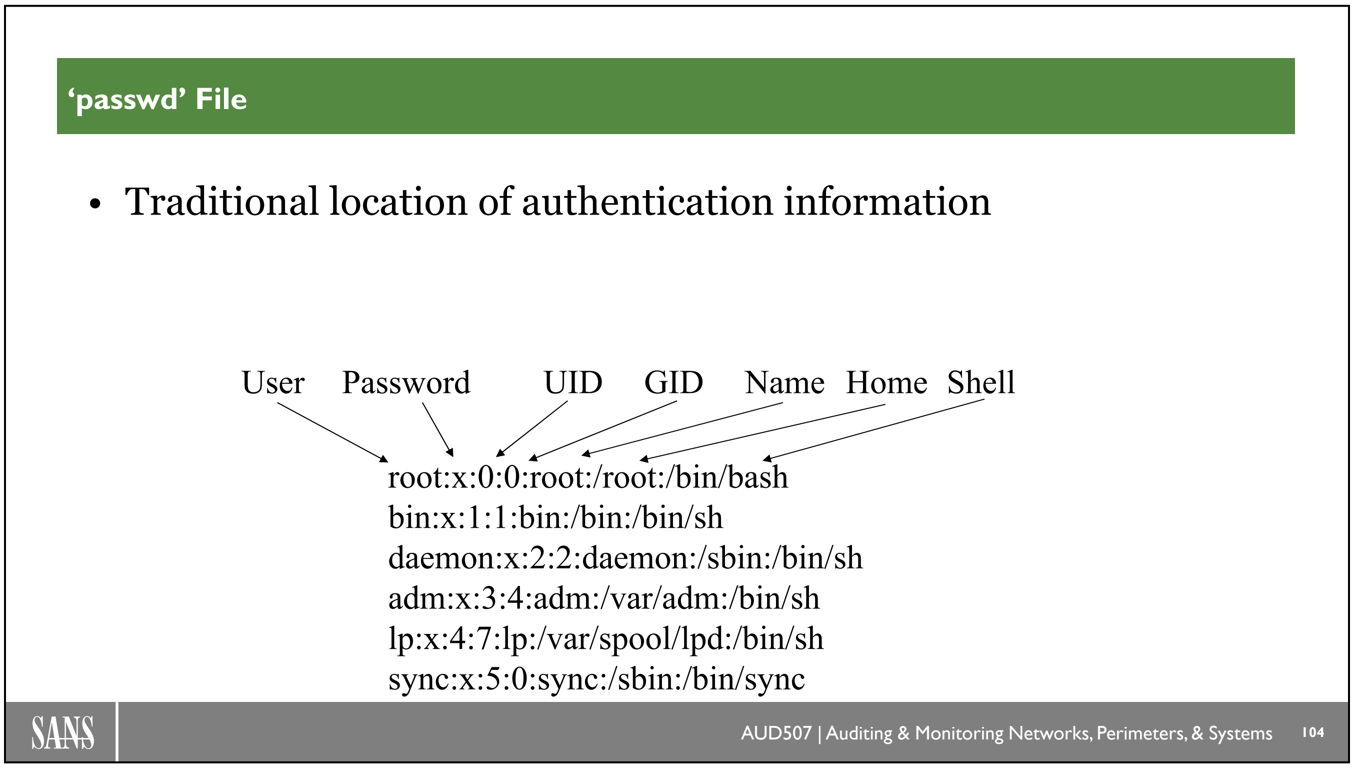
AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 103

In this section we discuss the issues you should consider when planning an audit of user account configuration, privileged access and remote access for your systems.

August 10, 2021

August 10, 2021

© SANS Institute 2021



‘passwd’ File

- Traditional location of authentication information

User
Password
UID
GID
Name
Home
Shell

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/sh
daemon:x:2:2:daemon:/sbin:/bin/sh
adm:x:3:4:adm:/var/adm:/bin/sh
lp:x:4:7:lp:/var/spool/lpd:/bin/sh
sync:x:5:0:sync:/sbin:/bin/sync
```

The password file is the traditional location of user credential information. The file is, by default, world readable. This is done to permit the system to correlate user IDs to usernames when looking at directory listings and process lists.

In the slide, we described each of the various fields in a typical password file. We begin with the username in the first column and the password in the second column. Notice that in this case all of the passwords are marked “x”. This column is actually where the encrypted password would reside. If this field is blank, then there is no password. Seeing all of these fields marked with an “x”, however, indicates that the encrypted password hashes are stored somewhere else, which we will see on the next slide. The next column is the actual user ID number followed by the primary group ID. Next, we have the “real name” or more descriptive name for the user, and then the home directory into which the user is placed when he logs in, and, finally, the default shell that is run when the user logs in.

It is very important to note that the presence of the “x” in the second field is *critical.* If this field is empty, then there is no password required for the user, regardless of what might be configured in the shadow file.

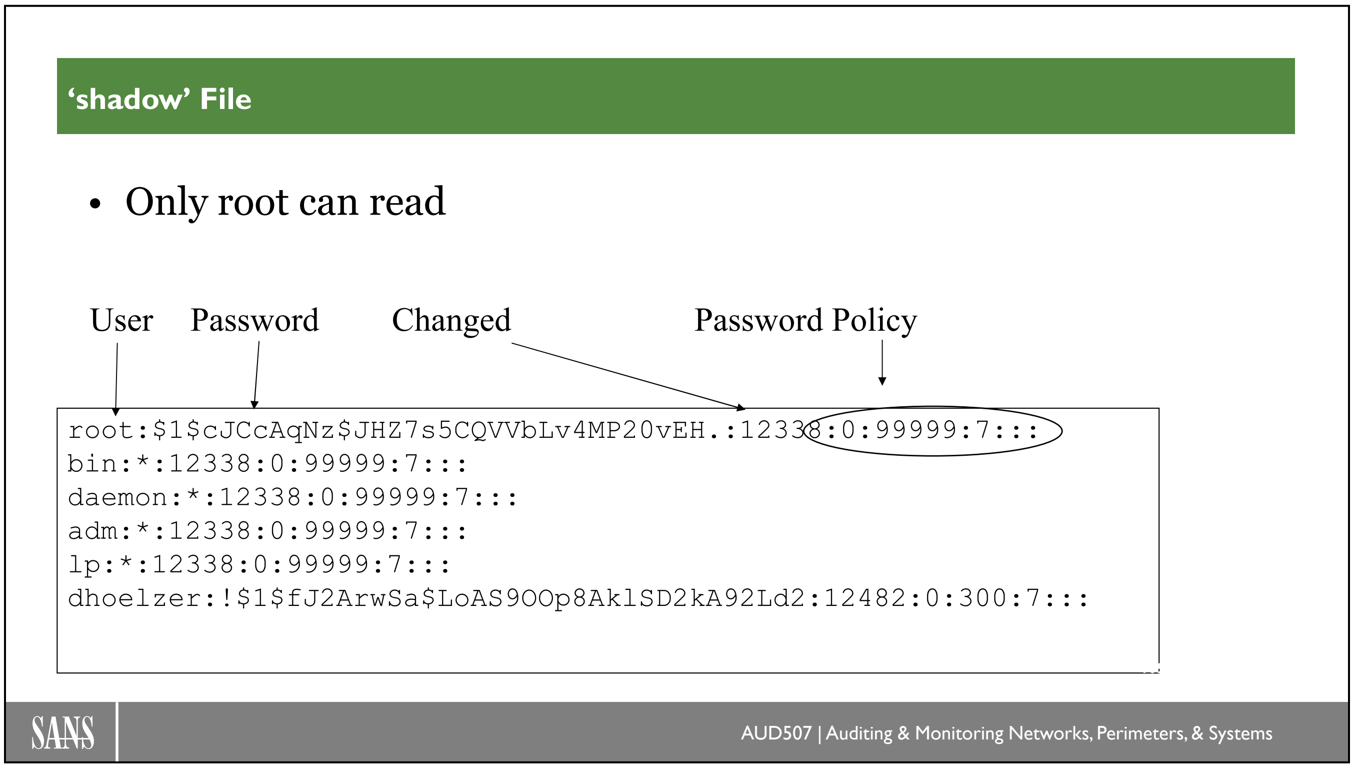
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



'shadow' File

- Only root can read

User
Password
Changed
Password Policy

```
root:$1$cJCcAqNz$JHZ7s5CQVVbLv4MP20vEH.:12338:0:99999:7:::
bin:*:12338:0:99999:7:::
daemon:*:12338:0:99999:7:::
adm:*:12338:0:99999:7:::
lp:*:12338:0:99999:7:::
dhoelzer:!$1$fJ2ArwSa$LoAS9OOp8AklSD2kA92Ld2:12482:0:300:7:::
```

The shadow password system was created first to eliminate the readability of encrypted password hashes and second to provide alternative means of encrypting the passwords. In this case, for instance, the encrypted password hash is an MD5-style password hash rather than the typical DES hash. How can we tell? The "$1$" is used to indicate the hashing type in use. Valid values today are:

1: MD5

2: Blowfish (anything starting with a 2 is a bcrypt variation)

3: NT Hash

5: SHA-256

6: SHA-512

The next value in the password string between the second and third dollar signs (cJCcAqNz) is the random salt that was added to the password to generate the hash.

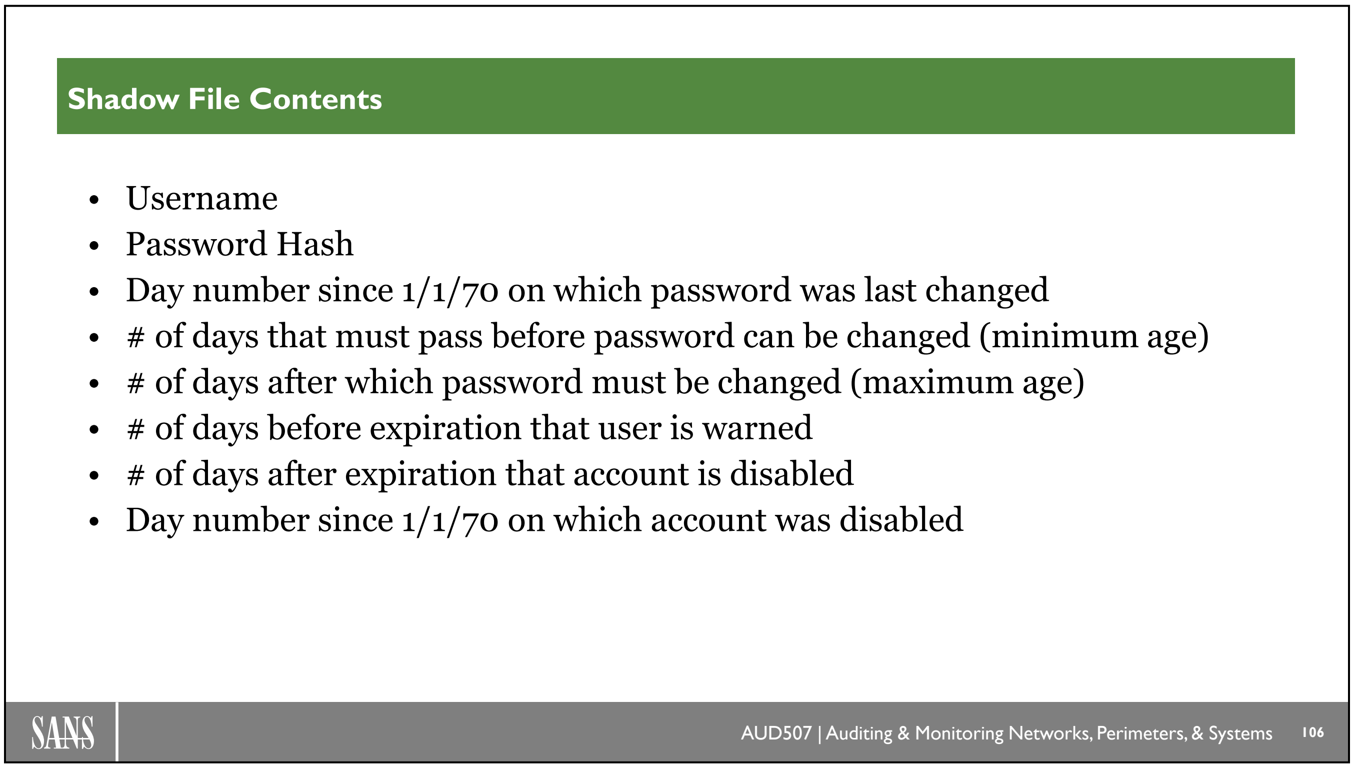
Notice that many of the password fields have an asterisk (*) in them. Just as in the passwd file, if the field is blank, it indicates no password, but any other character represents a password. What is the significance of a single character? Well, because the password hash function will always create the same number of characters (definitely > 1), this represents a password that can never be matched. In other words, the account is not disabled, but it is impossible to log in to the account directly!

Other additions to the typical passwd file are the inclusion of the post-epoch day on which the password was last changed, followed by the password policy information, which includes the minimum password age, maximum password age, expiration warning timer, post-expiration disable timer, and a count of how many days an account has been disabled. Quite honestly, very few people, including administrators, actually know how to interpret a shadow file, so you're already a leg up on them!

August 10, 2021

August 10, 2021

© SANS Institute 2021



Shadow File Contents

- Username
- Password Hash
- Day number since 1/1/70 on which password was last changed
- # of days that must pass before password can be changed (minimum age)
- # of days after which password must be changed (maximum age)
- # of days before expiration that user is warned
- # of days after expiration that account is disabled
- Day number since 1/1/70 on which account was disabled

As you can see from the slide, there are a number of fields in the shadow file that can be used to configure password controls for user accounts. In our experience, however, UNIX systems are typically configured to allow zero days between changes and 99,999 days before a change is required, which effectively turns off password policies.

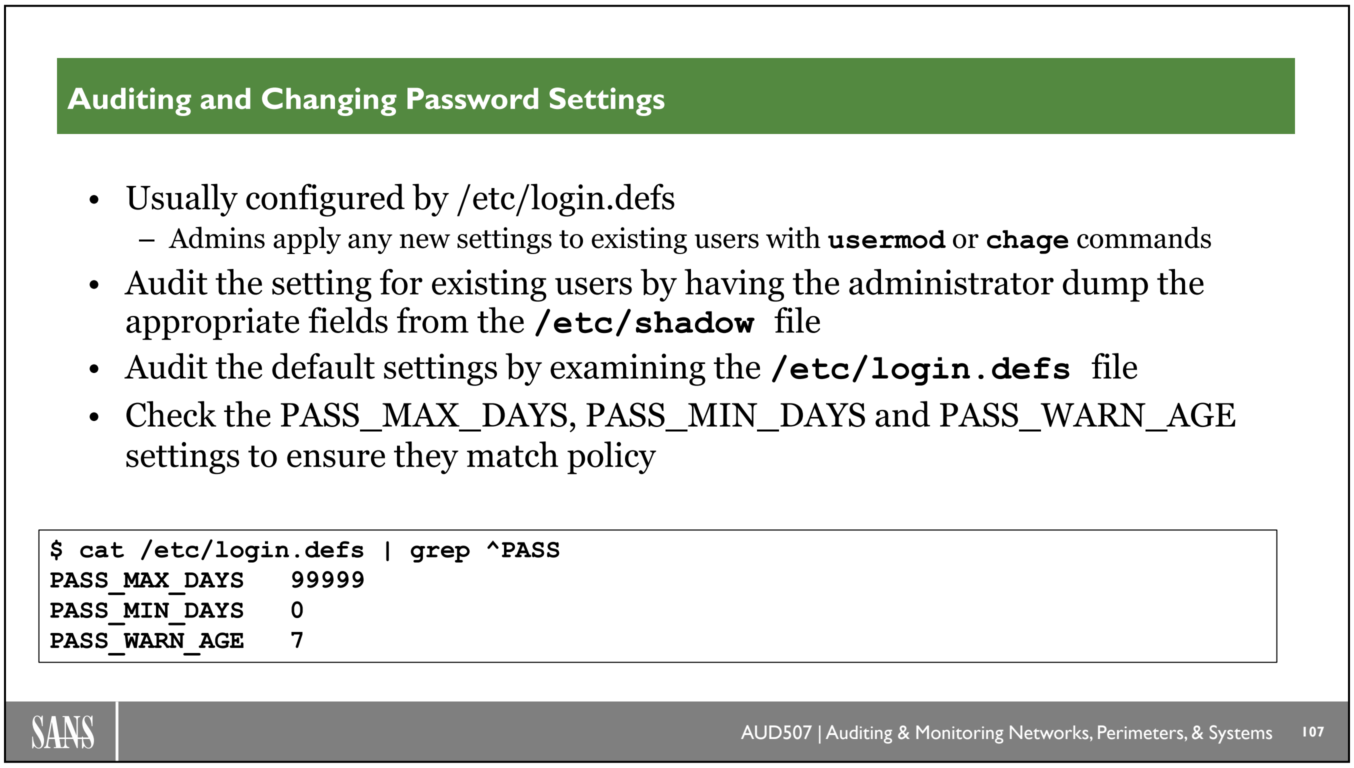
It is also useful to know that even after the account has been disabled, it is possible for "cron" jobs and "at" jobs to continue to run with this account's credentials. In fact, it is usually still possible to use the account to ftp into the system remotely and even to use the "su" tool to change to this user ID. Disabling the account effectively prevents only console-based access to the system.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Auditing and Changing Password Settings

- Usually configured by /etc/login.defs
  - Admins apply any new settings to existing users with **`usermod`** or **`chage`** commands
- Audit the setting for existing users by having the administrator dump the appropriate fields from the **`/etc/shadow`** file
- Audit the default settings by examining the **`/etc/login.defs`** file
- Check the PASS_MAX_DAYS, PASS_MIN_DAYS and PASS_WARN_AGE settings to ensure they match policy

```
$ cat /etc/login.defs | grep ^PASS
PASS_MAX_DAYS    99999
PASS_MIN_DAYS    0
PASS_WARN_AGE    7
```

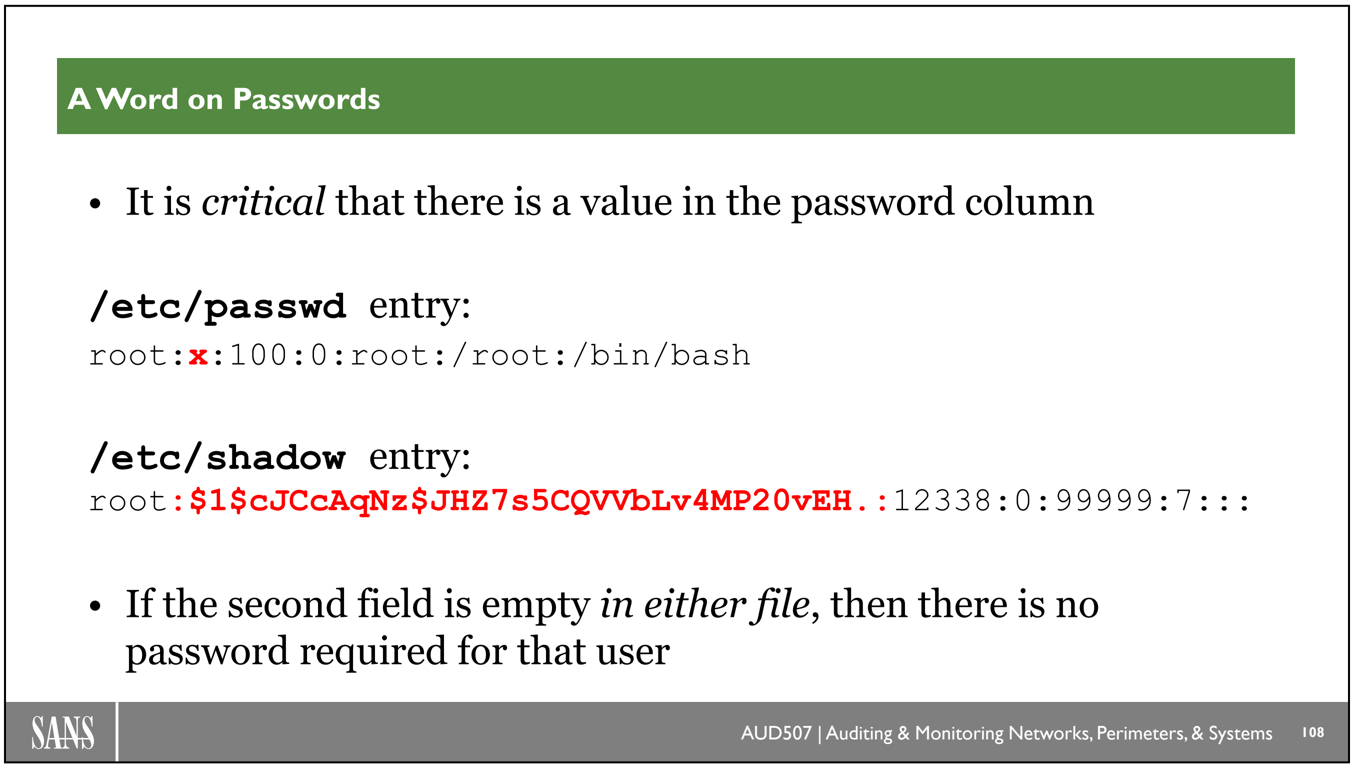
SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 107

If your organization requires password change controls, you will want to look not only at the shadow file itself for evidence that the settings are correct but also at the configuration file for the useradd tool (or whichever tool is used in your environment for the creation and management of user IDs on your UNIX system). Typically, this is found on Linux in the /etc/login.defs file. Please note that if an administrator changes the default settings, it has no impact on the existing accounts! They must be modified manually using the **`chage`** command to apply individual settings.

August 10, 2021

August 10, 2021

© SANS Institute 2021



A Word on Passwords

- It is *critical* that there is a value in the password column

/etc/passwd entry:

```
root:x:100:0:root:/root:/bin/bash
```

/etc/shadow entry:

```
root:$1$cJCcAqNz$JHZ7s5CQVVbLv4MP20vEH.:12338:0:99999:7:::
```

- If the second field is empty *in either file*, then there is no password required for that user

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 108

Remember that we are interested in identifying any operating system–specific vulnerabilities in our research. There is just such a vulnerability that is very specific to UNIX password files.

Typically, the second column in the colon-separated /etc/passwd or /etc/shadow file contains the user's password hash. What happens when this field is empty? It means that there is no password required for that user.

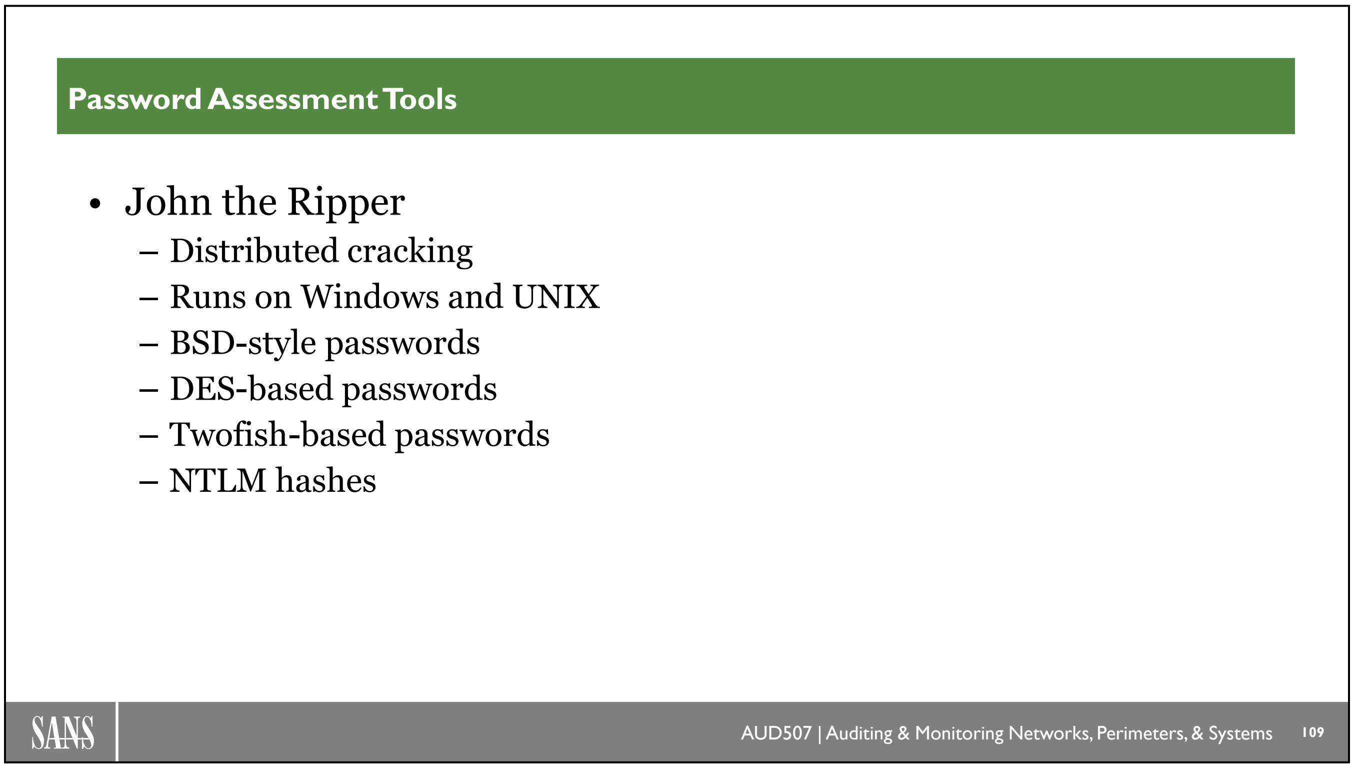
This presents a very interesting situation when the shadow password system is in use. If the entry in /etc/passwd is missing a value, even if there is a hash in the /etc/shadow file, *there is no password required for that user!* What makes this particularly dangerous is that if an administrator attempts to authenticate with a password across the network using something like SSH, the password that generates the hash in the /etc/shadow file will still work; it's simply not required. This can make it more difficult for the administrator to detect through normal use that the change has been made.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Password Assessment Tools

- John the Ripper
  - Distributed cracking
  - Runs on Windows and UNIX
  - BSD-style passwords
  - DES-based passwords
  - Twofish-based passwords
  - NTLM hashes

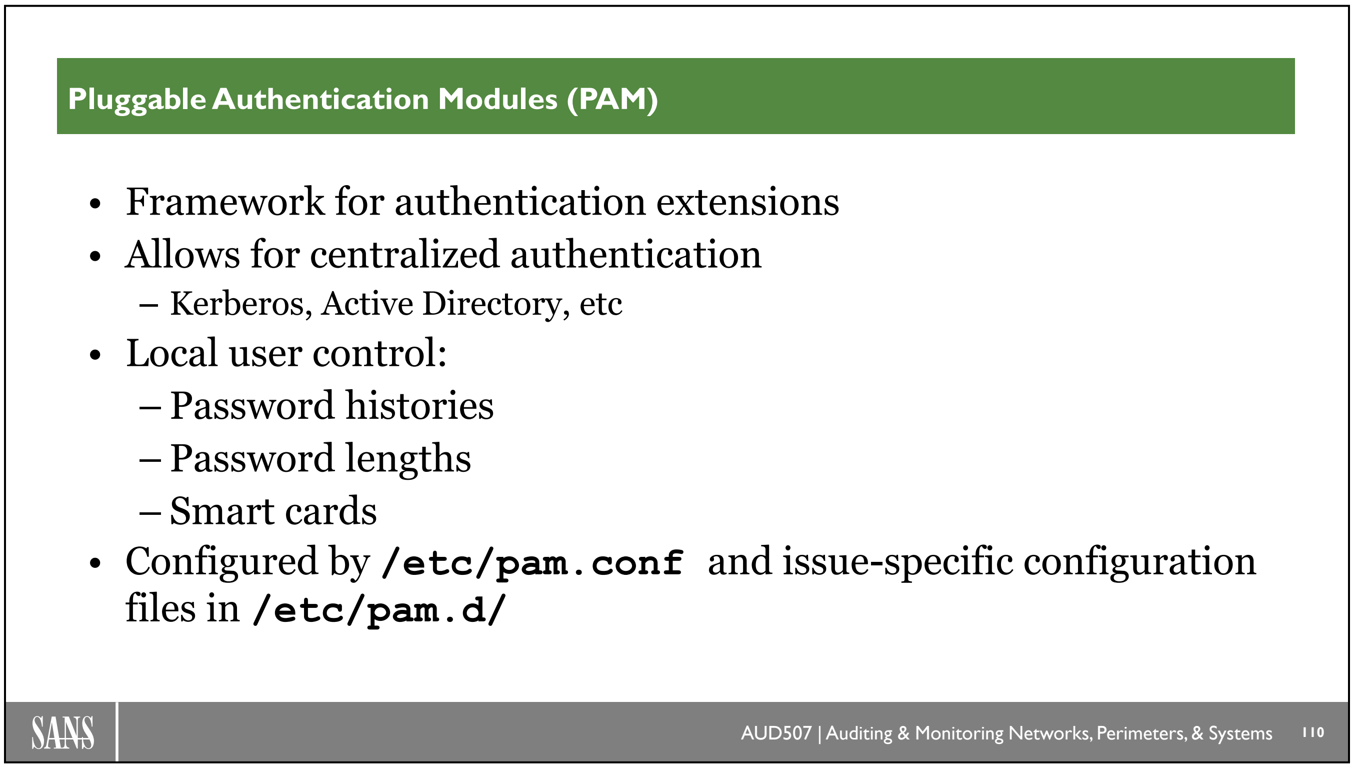
Of course, no audit would be complete without running a password assessment. One of the SANS Top Ten Vulnerabilities was "User Accounts with No or Weak Passwords." We recommend using a password assessment tool randomly and in conjunction with password filters that permit only "strong" passwords. The threat of cracking is usually enough to make people choose good passwords in the first place.

We give you a UNIX-based tool that not only can break Windows passwords, but can also run on Windows computers! In fact, this tool also allows you to perform distributed password-cracking attacks, speeding up the entire process. John is also capable, through add-on modules, of breaking just about any kind of password hash that you'll come across, making it a pretty versatile tool.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Pluggable Authentication Modules (PAM)

- Framework for authentication extensions
- Allows for centralized authentication
  - Kerberos, Active Directory, etc
- Local user control:
  - Password histories
  - Password lengths
  - Smart cards
- Configured by **`/etc/pam.conf`** and issue-specific configuration files in **`/etc/pam.d/`**

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 110

As an alternative or supplement to the traditional passwd/shadow files on our UNIX systems, we can leverage a product called Pluggable Authentication Modules (PAM). This is a framework that provides various authentication modules and controls that can be applied to our systems. Among other things, PAM can be used to configure our UNIX systems to use Active Directory credentials for users logging on to the systems for administrative or other purposes. This is a very powerful feature because it not only allows us to reuse a trusted credential store (good security principle), but it also means that the password policies we are enforcing in our Active Directory are now also being applied to our UNIX systems.

Consider the side effects. For example, password resets are now being *enforced*. Password complexity is now consistent across the organization. Password histories, which are otherwise unavailable in UNIX, can now be enforced. Perhaps the best feature, though, is that by connecting our authentication to our Active Directory, we need to disable a user in *only one place!*

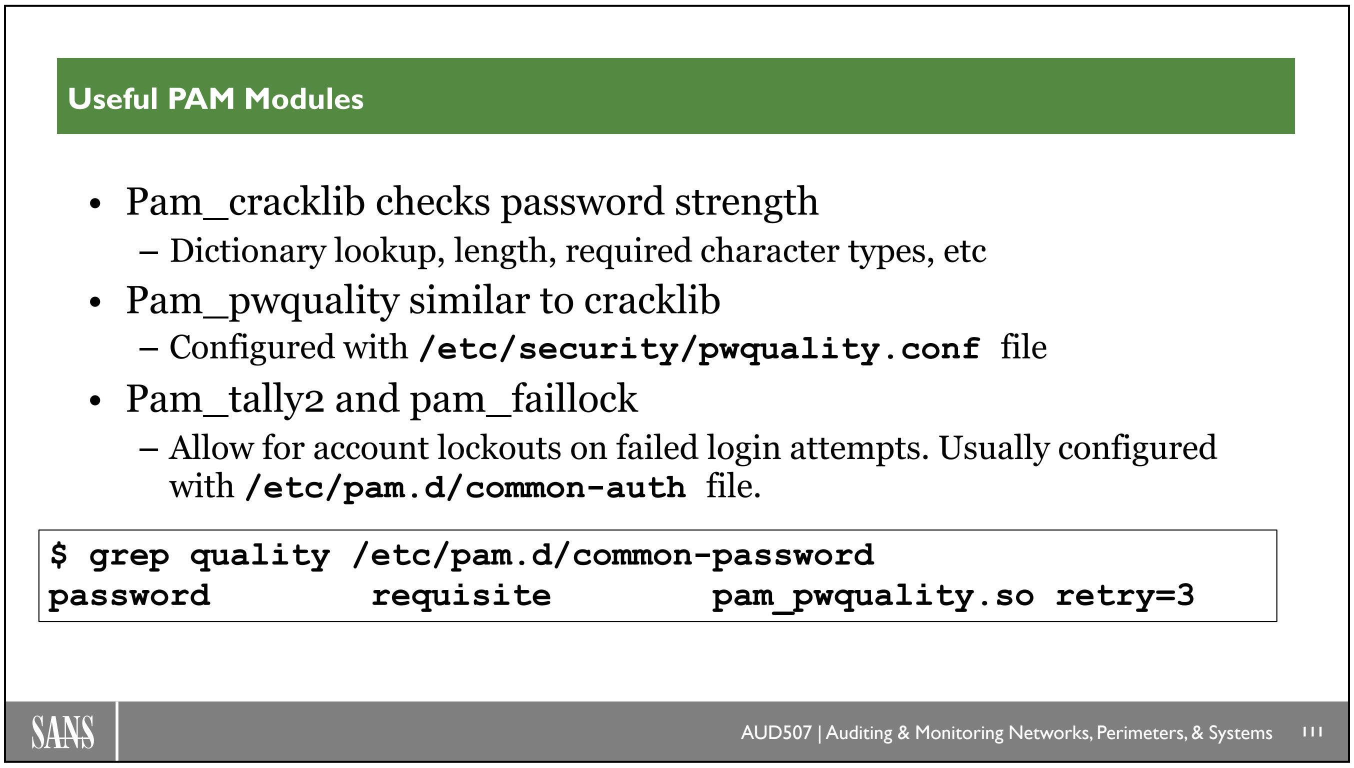
The main configuration file for PAM is usually **`/etc/pam.conf`**, supplemented by module-specific configuration files in the **`/etc/pam.d`** directory.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Useful PAM Modules

- Pam_cracklib checks password strength
  - Dictionary lookup, length, required character types, etc
- Pam_pwquality similar to cracklib
  - Configured with **`/etc/security/pwquality.conf`** file
- Pam_tally2 and pam_faillock
  - Allow for account lockouts on failed login attempts. Usually configured with **`/etc/pam.d/common-auth`** file.

```
$ grep quality /etc/pam.d/common-password
password        requisite        pam_pwquality.so retry=3
```

This slide lists some useful PAM modules that you might encounter during your audits.

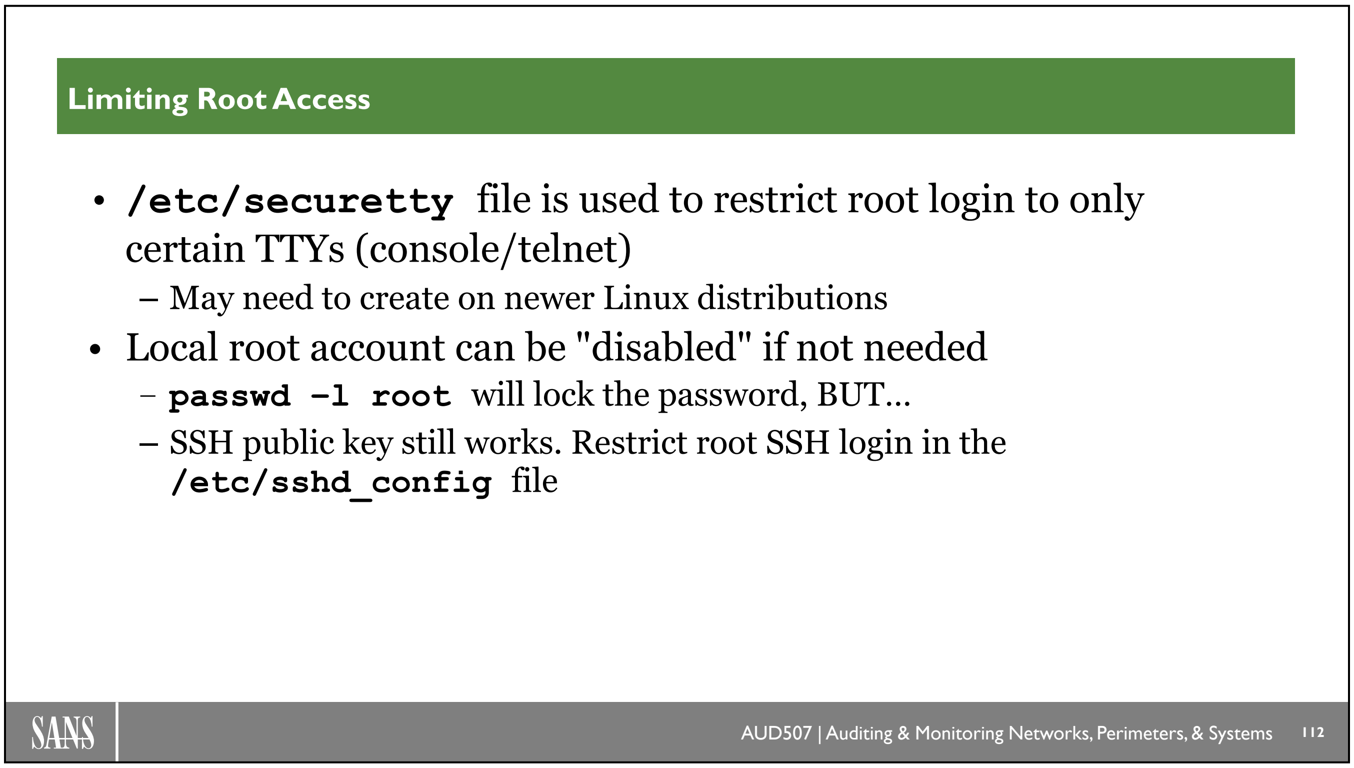
Pam_cracklib and Pam_pwquality are both used to check the quality of new passwords before allowing them to be created. Pam_pwquality is configured using the settings in the **`/etc/security/pwquality.conf`** file.

Pam_tally2 and pam_faillock are used to add account lockout capabilities to the system. The settings for the number of failed attempts required to lockout an account and the default lockout time period are usually found in the **`/etc/pam.d/common-auth`** file.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Limiting Root Access

- **`/etc/securetty`** file is used to restrict root login to only certain TTYs (console/telnet)
  - May need to create on newer Linux distributions
- Local root account can be "disabled" if not needed
  - **`passwd -l root`** will lock the password, BUT…
  - SSH public key still works. Restrict root SSH login in the **`/etc/sshd_config`** file

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 112

Another item that falls into this section is controls that limit the access to certain accounts from certain places. For instance, should users be able to log in to the root account directly across the network or would you prefer that they first authenticate as some other user? To allow for some control here, most UNIX systems allow this to be controlled through the use of the /etc/securetty file. This file contains a list of all the ttys (teletypes… yes, we know that we don't use teletypes anymore, but UNIX has been around for a long time—think of these as endpoints) that are considered secure enough to permit the root user to log in to them.

While telnet should not be in use on our hosts any longer, we should still verify that the only terminals that permit direct logins by the root user are physically connected to the server. These days, that usually means that they are at the console itself.

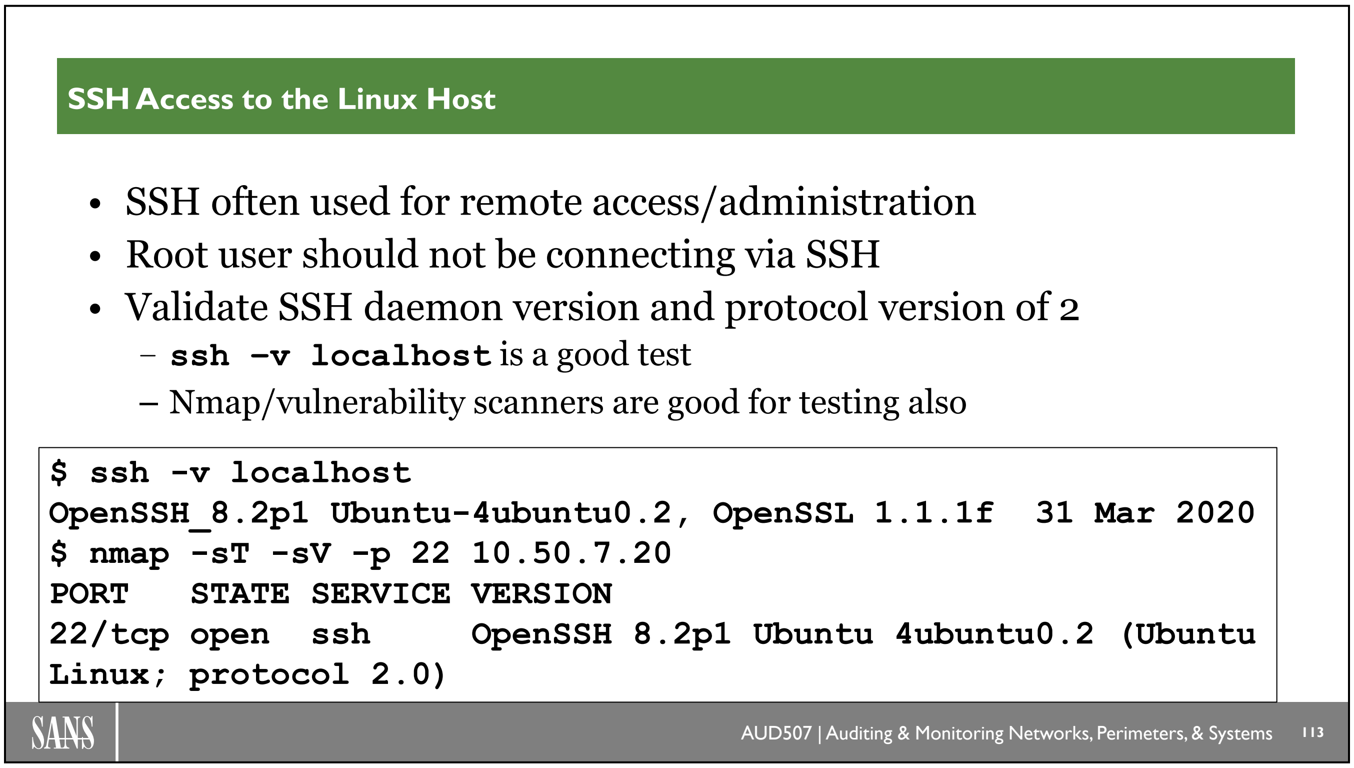
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



SSH Access to the Linux Host

- SSH often used for remote access/administration
- Root user should not be connecting via SSH
- Validate SSH daemon version and protocol version of 2
  - **`ssh -v localhost`** is a good test
  - Nmap/vulnerability scanners are good for testing also

```
$ ssh -v localhost
OpenSSH_8.2p1 Ubuntu-4ubuntu0.2, OpenSSL 1.1.1f  31 Mar 2020
$ nmap -sT -sV -p 22 10.50.7.20
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu
Linux; protocol 2.0)
```

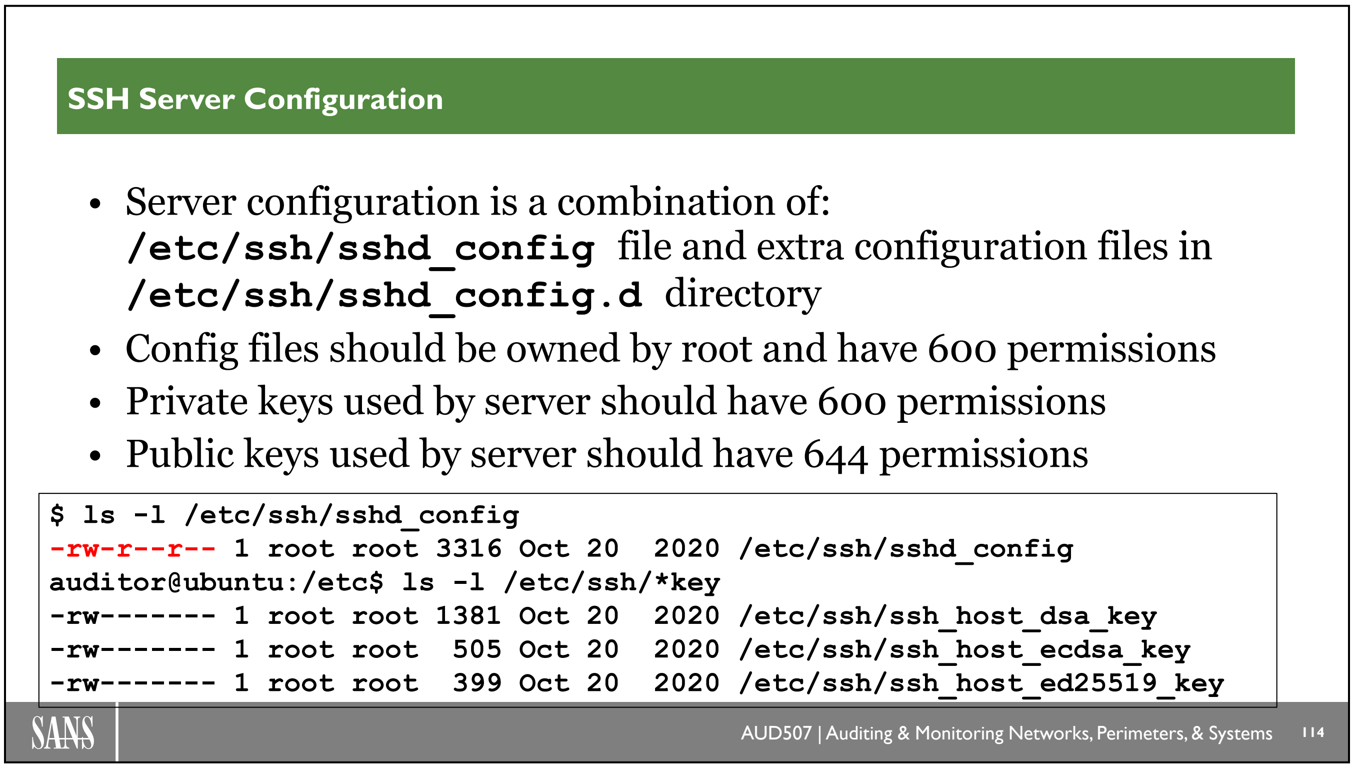
Today, most administration and remote access to our Unix systems is done via secure shell (SSH). While we may keep our local root account enabled and allowed to logon to the console for "break-glass" types of emergencies, there is really no good reason for root to login via SSH. Instead, the administrators should logon as themselves using SSH and only elevate their privileges when necessary. This makes accounting for administrators' actions much easier.

Since SSH has had some issues with the protocol over the years, it's import to make sure the host under audit is running the current protocol, and that the SSH software is the current version.

August 10, 2021

August 10, 2021

© SANS Institute 2021



SSH Server Configuration

- Server configuration is a combination of: **`/etc/ssh/sshd_config`** file and extra configuration files in **`/etc/ssh/sshd_config.d`** directory
- Config files should be owned by root and have 600 permissions
- Private keys used by server should have 600 permissions
- Public keys used by server should have 644 permissions

```
$ ls -l /etc/ssh/sshd_config
-rw-r--r-- 1 root root 3316 Oct 20  2020 /etc/ssh/sshd_config
auditor@ubuntu:/etc$ ls -l /etc/ssh/*key
-rw------- 1 root root 1381 Oct 20  2020 /etc/ssh/ssh_host_dsa_key
-rw------- 1 root root  505 Oct 20  2020 /etc/ssh/ssh_host_ecdsa_key
-rw------- 1 root root  399 Oct 20  2020 /etc/ssh/ssh_host_ed25519_key
```

The security of an SSH server is largely based on the security of the SSH daemon itself and the user environment configuration. Keys and configuration files used by the server should be owned by root and have appropriate permissions applied, On this slide we give example of what some of those files and permissions might be. Please remember, though, that there may be dozens of files which need to be included in your organization's hardening standards.

Note that in the first command in the screenshot, the permissions for the sshd_config file are too permissive.

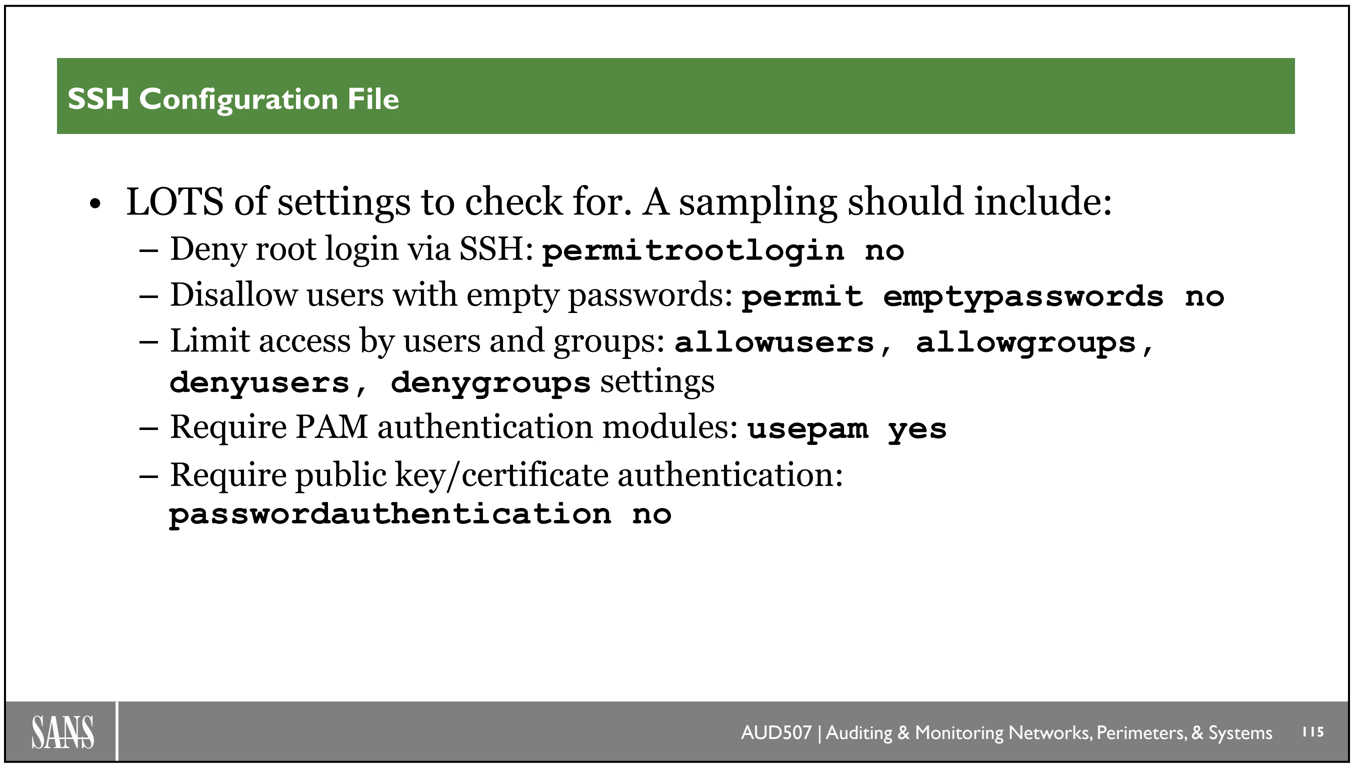
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



SSH Configuration File

- LOTS of settings to check for. A sampling should include:
  - Deny root login via SSH: **`permitrootlogin no`**
  - Disallow users with empty passwords: **`permit emptypasswords no`**
  - Limit access by users and groups: **`allowusers, allowgroups, denyusers, denygroups`** settings
  - Require PAM authentication modules: **`usepam yes`**
  - Require public key/certificate authentication: **`passwordauthentication no`**

SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 115

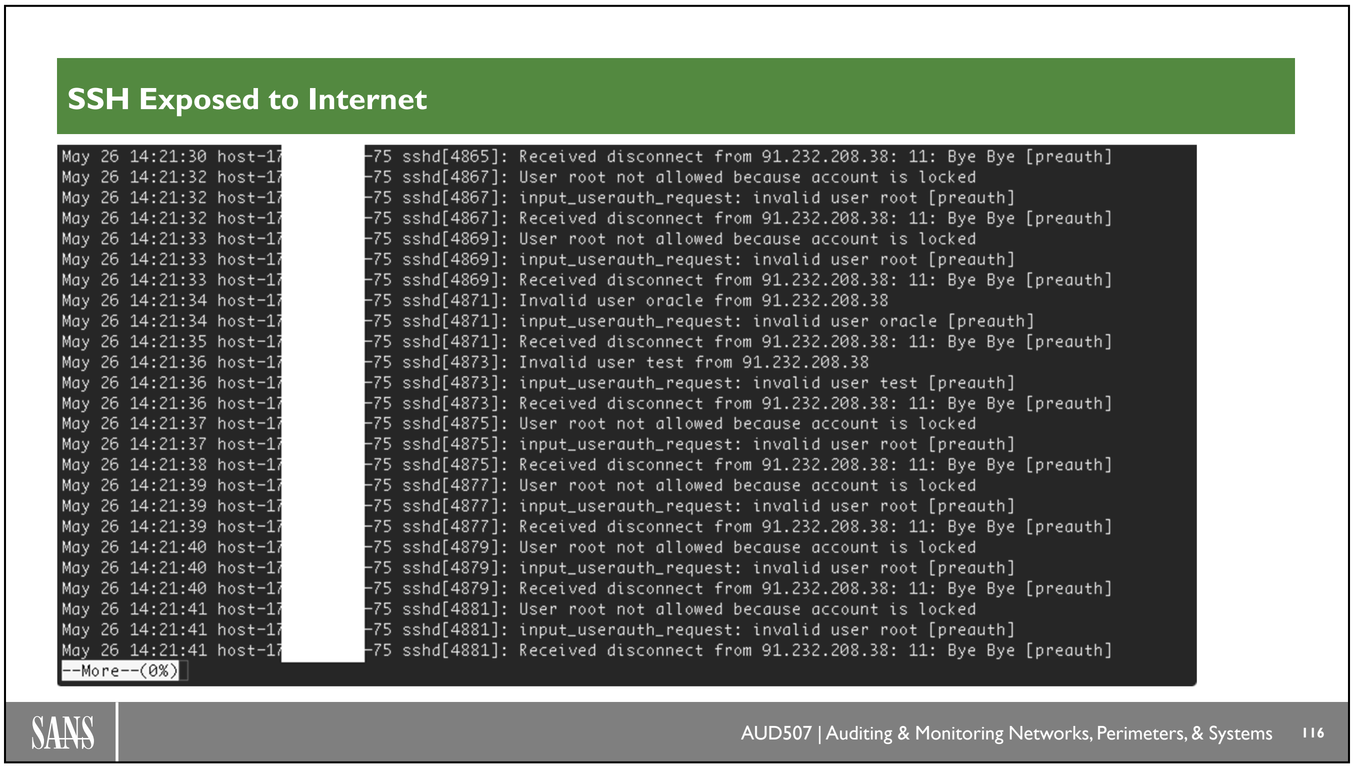
Access to the system should be restricted. Simply installing and turning on SSH will allow any valid user to authenticate. Instead, only specific users or, preferably, groups should be permitted to use SSH.

As mentioned earlier, the root user should not be allowed to login via SSH. Neither should users who have an empty local password in the shadow file. Other user access should be restricted with the allow/deny users/groups settings. If your organization uses PAM modules to enhance authentication, check to see that the PAM modules are being required for SSH, also. For those individuals who are permitted access, you would like to find that only public keys or certificates are used for authentication, never passwords. Why is that? Take a look at the next slide…

August 10, 2021

August 10, 2021

© SANS Institute 2021



SSH Exposed to Internet

```
May 26 14:21:30 host-17[redacted]-75 sshd[4865]: Received disconnect from 91.232.208.38: 11: Bye Bye [preauth]
May 26 14:21:32 host-17[redacted]-75 sshd[4867]: User root not allowed because account is locked
May 26 14:21:32 host-17[redacted]-75 sshd[4867]: input_userauth_request: invalid user root [preauth]
May 26 14:21:32 host-17[redacted]-75 sshd[4867]: Received disconnect from 91.232.208.38: 11: Bye Bye [preauth]
May 26 14:21:33 host-17[redacted]-75 sshd[4869]: User root not allowed because account is locked
May 26 14:21:33 host-17[redacted]-75 sshd[4869]: input_userauth_request: invalid user root [preauth]
May 26 14:21:33 host-17[redacted]-75 sshd[4869]: Received disconnect from 91.232.208.38: 11: Bye Bye [preauth]
May 26 14:21:34 host-17[redacted]-75 sshd[4871]: Invalid user oracle from 91.232.208.38
May 26 14:21:34 host-17[redacted]-75 sshd[4871]: input_userauth_request: invalid user oracle [preauth]
May 26 14:21:35 host-17[redacted]-75 sshd[4871]: Received disconnect from 91.232.208.38: 11: Bye Bye [preauth]
May 26 14:21:36 host-17[redacted]-75 sshd[4873]: Invalid user test from 91.232.208.38
May 26 14:21:36 host-17[redacted]-75 sshd[4873]: input_userauth_request: invalid user test [preauth]
May 26 14:21:36 host-17[redacted]-75 sshd[4873]: Received disconnect from 91.232.208.38: 11: Bye Bye [preauth]
May 26 14:21:37 host-17[redacted]-75 sshd[4875]: User root not allowed because account is locked
May 26 14:21:37 host-17[redacted]-75 sshd[4875]: input_userauth_request: invalid user root [preauth]
May 26 14:21:38 host-17[redacted]-75 sshd[4875]: Received disconnect from 91.232.208.38: 11: Bye Bye [preauth]
May 26 14:21:39 host-17[redacted]-75 sshd[4877]: User root not allowed because account is locked
May 26 14:21:39 host-17[redacted]-75 sshd[4877]: input_userauth_request: invalid user root [preauth]
May 26 14:21:39 host-17[redacted]-75 sshd[4877]: Received disconnect from 91.232.208.38: 11: Bye Bye [preauth]
May 26 14:21:40 host-17[redacted]-75 sshd[4879]: User root not allowed because account is locked
May 26 14:21:40 host-17[redacted]-75 sshd[4879]: input_userauth_request: invalid user root [preauth]
May 26 14:21:40 host-17[redacted]-75 sshd[4879]: Received disconnect from 91.232.208.38: 11: Bye Bye [preauth]
May 26 14:21:41 host-17[redacted]-75 sshd[4881]: User root not allowed because account is locked
May 26 14:21:41 host-17[redacted]-75 sshd[4881]: input_userauth_request: invalid user root [preauth]
May 26 14:21:41 host-17[redacted]-75 sshd[4881]: Received disconnect from 91.232.208.38: 11: Bye Bye [preauth]
--More--(0%)
```

If you have a system running SSH connected to the internet, you should have a look at the logs. Unless you are restricting access to the SSH service based on a firewall rule, you will find tens to hundreds of thousands of login attempts to common accounts.

When you first see this, you might wonder why attackers would do this. Do they really expect to find accounts exposed with weak passwords? To answer that question, consider this: Why would a large number of attackers perform this same type of scan? The only reasonable answer is that they are meeting with success! Systems with default accounts and default passwords are rampant. Systems with weak passwords are even more common.

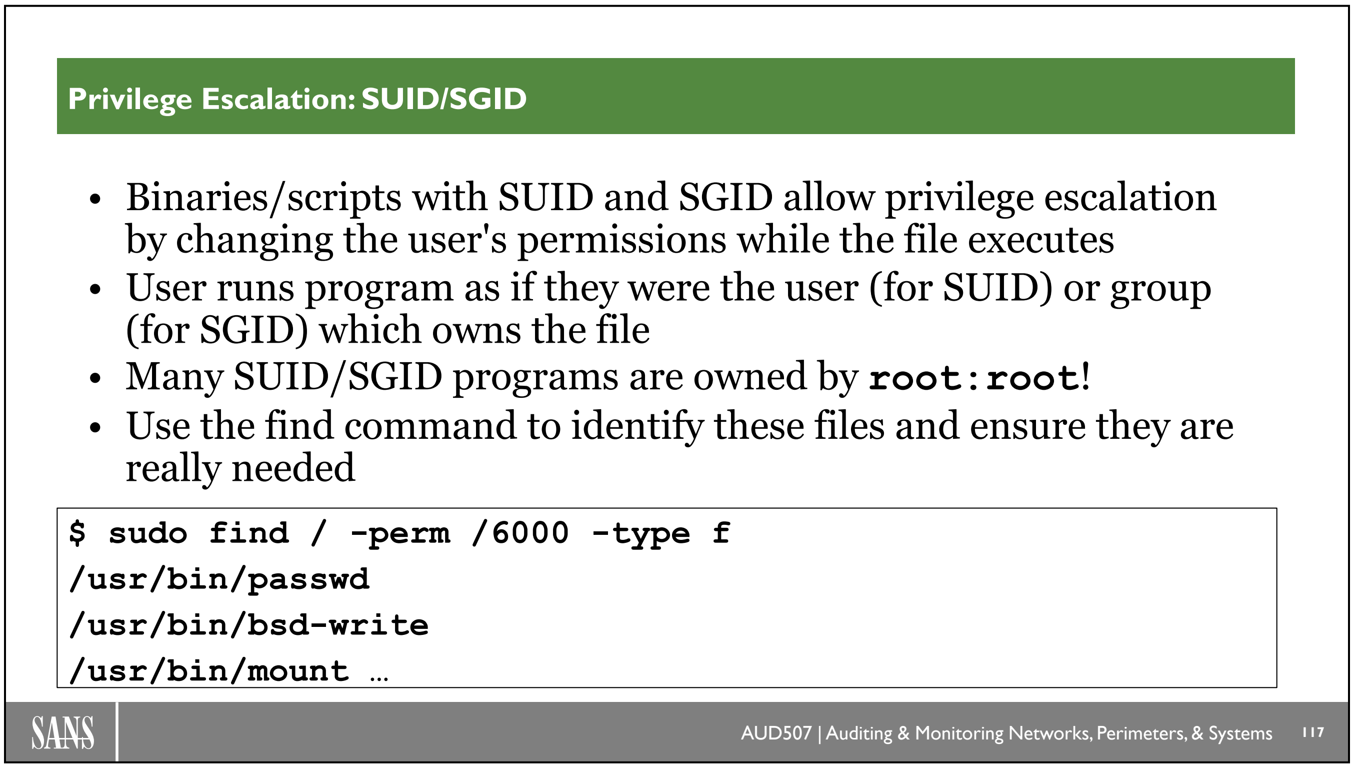
Requiring key- or certificate-based authentication eliminates this exposure.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Privilege Escalation: SUID/SGID

- Binaries/scripts with SUID and SGID allow privilege escalation by changing the user's permissions while the file executes
- User runs program as if they were the user (for SUID) or group (for SGID) which owns the file
- Many SUID/SGID programs are owned by **`root:root`**!
- Use the find command to identify these files and ensure they are really needed

```
$ sudo find / -perm /6000 -type f
/usr/bin/passwd
/usr/bin/bsd-write
/usr/bin/mount …
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 117

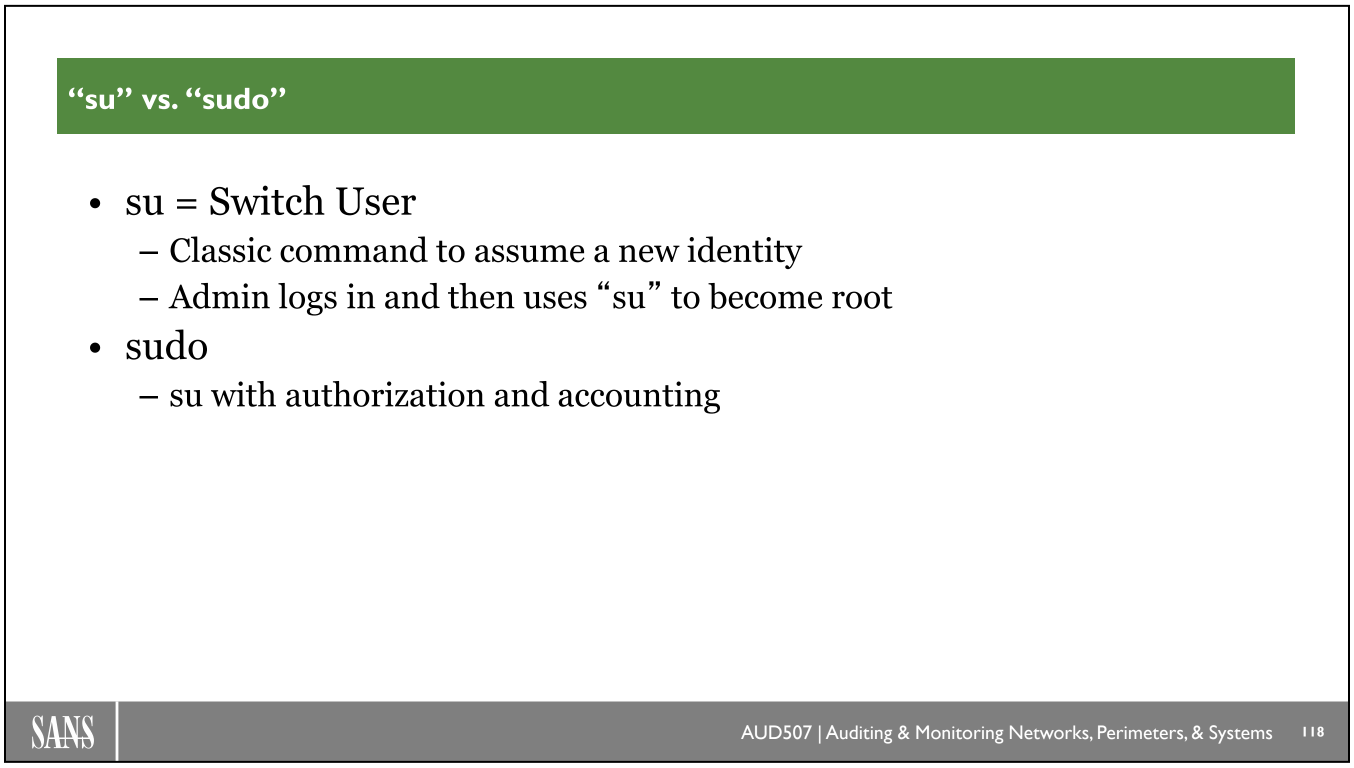
One form of privilege escalation on our systems can involve any user – not just administrators. On most Unixes, there are several binaries which have the SUID and/or SGID permission bits set. These bits are used to tell the system that the user should be given elevated privileges while running the program. Usually this is to allow the user to perform an action or update part of a file (the like shadow) file with their own information. The principle of least privilege tells us that we should minimize the number of SUID/SGID files on the system to only those that are required by the enterprise.

Audit for those files with the find command, which can search by exact permissions, or for any file with either the SUID or SGID flags, as we did in the example on the slide.

August 10, 2021

August 10, 2021

© SANS Institute 2021



"su" vs. "sudo"

- su = Switch User
  - Classic command to assume a new identity
  - Admin logs in and then uses "su" to become root
- sudo
  - su with authorization and accounting

During the course of this class, it has already been mentioned that administrators should never be permitted to log in directly as "root" or "administrator." Instead, they should be required to log in using a personal account and then become the administrator. The typical way that this has been done on UNIX systems over the years is using the "su" tool to "switch user."

A much better way to accomplish this is using a standard replacement for su: sudo. You can think of this command as meaning "su Do", which is what's actually happening. With sudo, the administrator logs in to the system using his personal account. When he wants to perform an administrative activity, he must enter "sudo <command>". For example, to kill a process as root, he could run, "sudo kill -9 12345".

The beautiful thing about this system is that the user can optionally be prompted for a password before the action is taken. The password that the user must enter is *his own* password, not the root password! Even better, every action taken using sudo is logged, creating useful accounting records. Finally, the system can be configured to create groups of users with different authorization requirements and permission to run specific tools in specific ways. You can even restrict which commands can be run. For example, you can allow a user to run "sudo su another_user" while preventing him from running "sudo su" or "sudo su root" or any variation thereof!

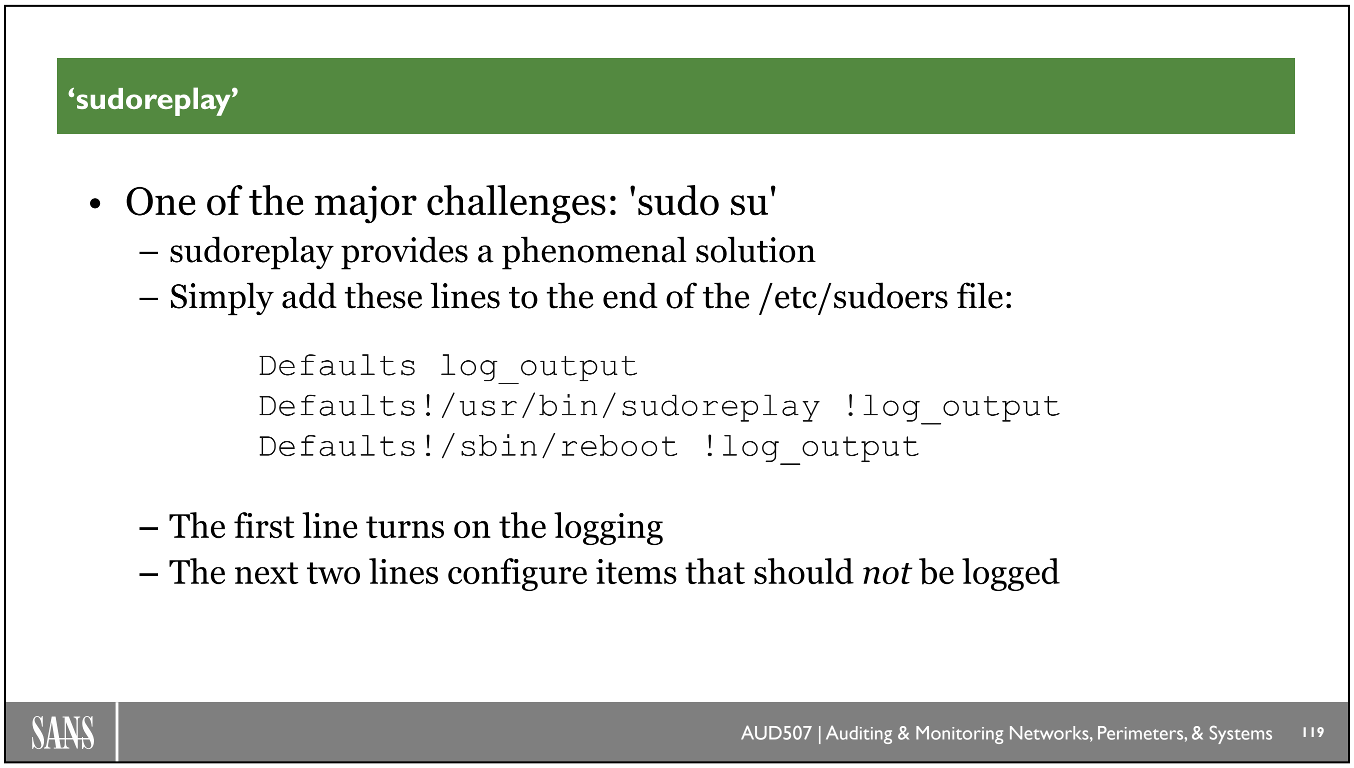
This tool is a must-use for administrators. Without something like this, there is no real way to account for activities on the system.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



'sudoreplay'

- One of the major challenges: 'sudo su'
  - sudoreplay provides a phenomenal solution
  - Simply add these lines to the end of the /etc/sudoers file:

```
Defaults log_output
Defaults!/usr/bin/sudoreplay !log_output
Defaults!/sbin/reboot !log_output
```

  - The first line turns on the logging
  - The next two lines configure items that should *not* be logged

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 119

'sudoreplay' is an extremely useful tool that, if configured, can be super for auditing administrative activities. Even with 'sudo' being used to track and audit administrator activities, it's a pretty simple matter for the administrator to either 'sudo su', becoming root, or escalate to root through some other means. With the addition of three simple lines to the /etc/sudoers file, however, the system will generate automatic audit trails for everything done under a sudo-spawned shell!

The first added configuration line is what actually sets up the logging. The following two lines use "!" (the "Not" character), indicating commands that should *not* be logged beyond an indicator that the commands were run.

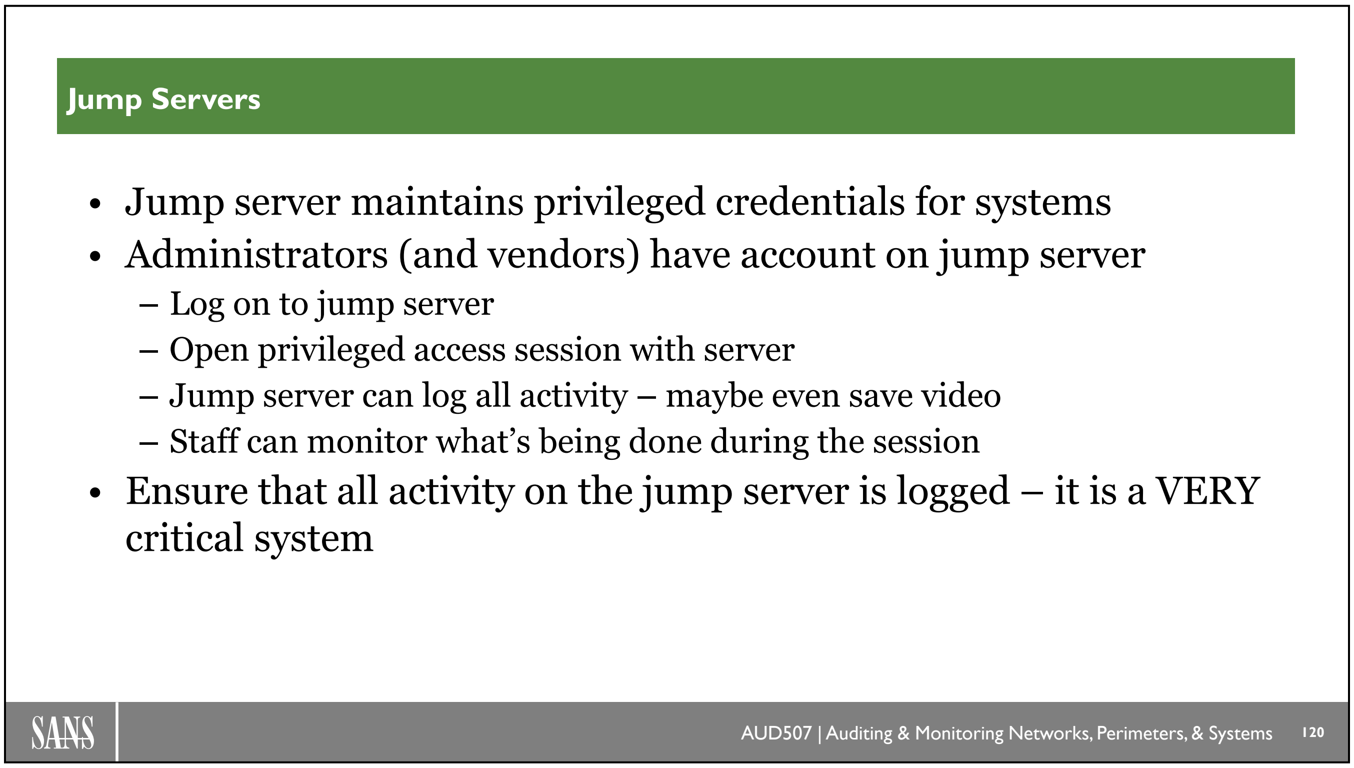
To view the replay, you can first execute 'sudoreplay –l' to obtain a list of all the sessions available. You can also provide a search expression, allowing you to search through all the transcripts of sessions for a particular command or user, for example. In the listing, you will find an "ID" that can then be used to replay that session.

When you now execute 'sudoreplay <ID>', the session will begin to replay in real-time! You can also press the spacebar to pause, the < key to halve the playback speed, or the > key to double playback speed! We have an example in the lab exercises. It's truly an amazing tool!

August 10, 2021

August 10, 2021

© SANS Institute 2021



Jump Servers

- Jump server maintains privileged credentials for systems
- Administrators (and vendors) have account on jump server
  - Log on to jump server
  - Open privileged access session with server
  - Jump server can log all activity – maybe even save video
  - Staff can monitor what's being done during the session
- Ensure that all activity on the jump server is logged – it is a VERY critical system

SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 120

The problem of auditing administrator access to systems is not a new one. The sudo tool provides a very realistic and inexpensive mechanism that can be implemented to begin accounting for these activities. The addition of sudoreplay makes it even more powerful. Even so, this is a solution only for our UNIX systems.

The idea of leveraging something like PAM to rely upon centralized authentication into an LDAP or perhaps our AD infrastructure also makes a great deal of sense from an enterprise perspective. This reduces the effort required to simply verify that the proper groups are being enforced in the directory and that the UNIX systems have been properly configured to enforce authentication requirements to the directory.

A larger-scale and more holistic solution does exist, however. Cyberark is an example of just such a tool. Although the price tag can be high, Cyberark provides a single tool that can be used to enforce administrative access controls across both your UNIX and Windows systems. It also provides an even better implementation of the sudoreplay concept, allowing you to record and later play back any and all administrative actions within your systems.

August 10, 2021

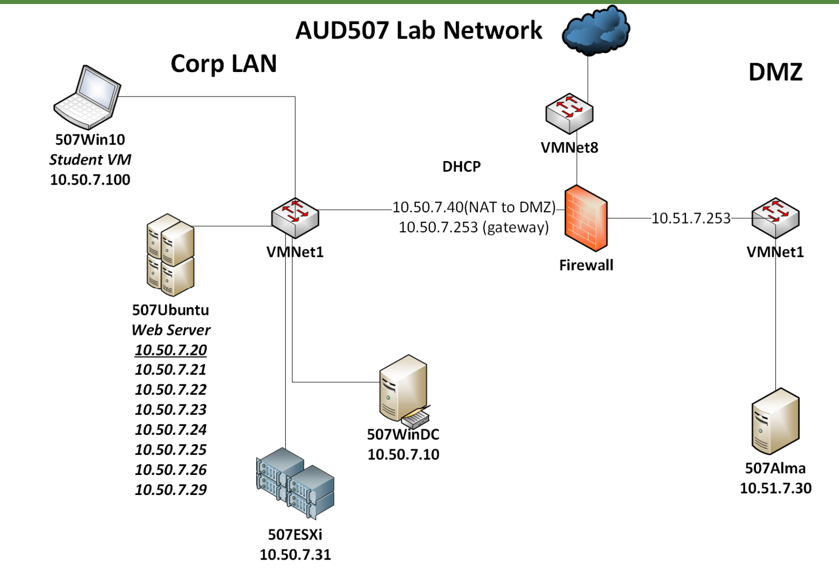
Using a jump server solution, none of the administrators actually know any of the administrative passwords. Instead, their connections are brokered through the jump server, which then creates the authenticated session using the actual credentials of the protected system.

August 10, 2021

Technet24

© SANS Institute 2021

Exercise 3.3: Services and Passwords

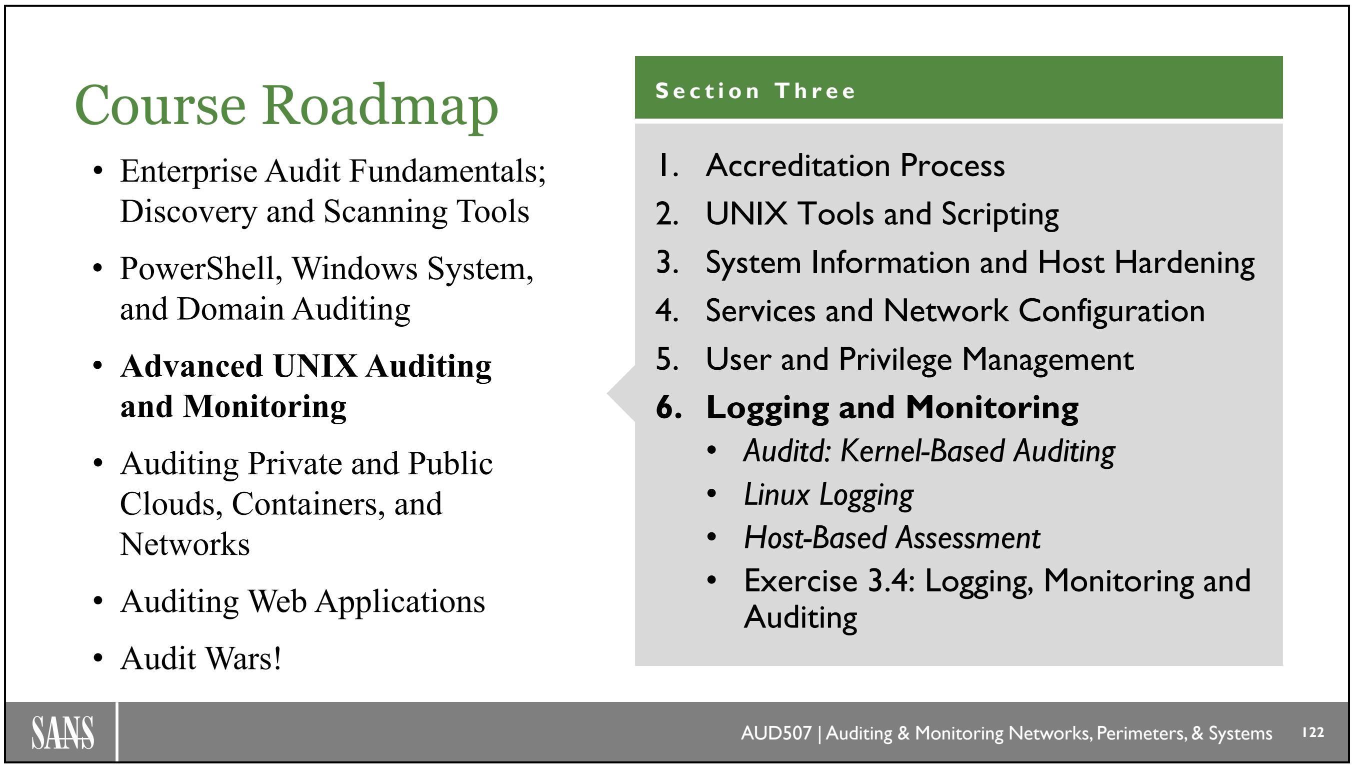


Please open your Workbook to Exercise 3.3: Services and Passwords or pick up with wherever you left off in the exercises.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Course Roadmap

- Enterprise Audit Fundamentals; Discovery and Scanning Tools
- PowerShell, Windows System, and Domain Auditing
- **Advanced UNIX Auditing and Monitoring**
- Auditing Private and Public Clouds, Containers, and Networks
- Auditing Web Applications
- Audit Wars!

Section Three

1. Accreditation Process
2. UNIX Tools and Scripting
3. System Information and Host Hardening
4. Services and Network Configuration
5. User and Privilege Management
6. **Logging and Monitoring**
   - *Auditd: Kernel-Based Auditing*
   - *Linux Logging*
   - *Host-Based Assessment*
   - Exercise 3.4: Logging, Monitoring and Auditing

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 122

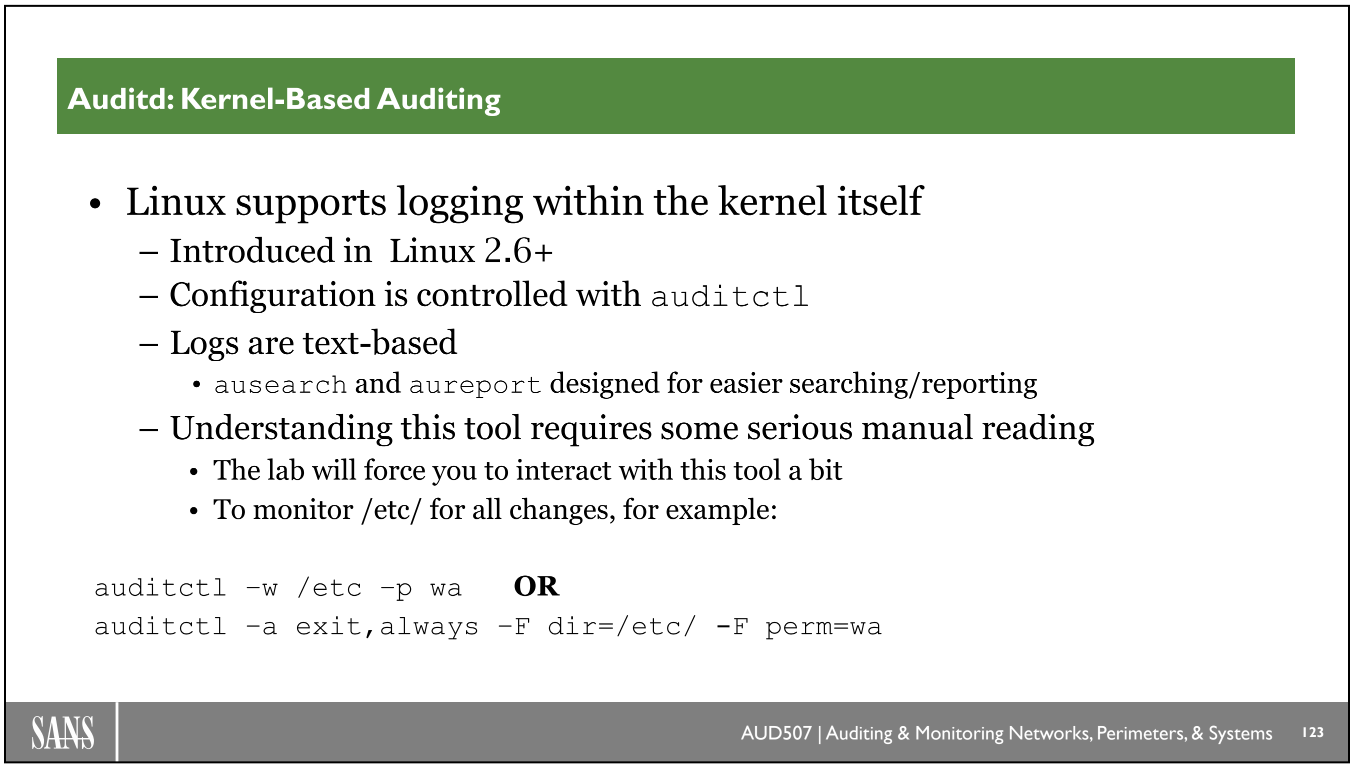
In this section, we will continue examining UNIX system elements and begin transitioning into system logging.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Auditd: Kernel-Based Auditing

- Linux supports logging within the kernel itself
  - Introduced in Linux 2.6+
  - Configuration is controlled with `auditctl`
  - Logs are text-based
    - `ausearch` and `aureport` designed for easier searching/reporting
  - Understanding this tool requires some serious manual reading
    - The lab will force you to interact with this tool a bit
    - To monitor /etc/ for all changes, for example:

```
auditctl -w /etc -p wa
```

OR

```
auditctl -a exit,always -F dir=/etc/ -F perm=wa
```

Linux systems with a kernel of version 2.6 or higher (modern systems) have an auditing facility built into the kernel itself. This system is configured using the auditctl command and creates text-based logs in the /var/log/audit directory.

Configuring the audit controls using this tool can be tricky. Rather than try to explain the complicated format here, we will examine it in some detail in the associated lab.

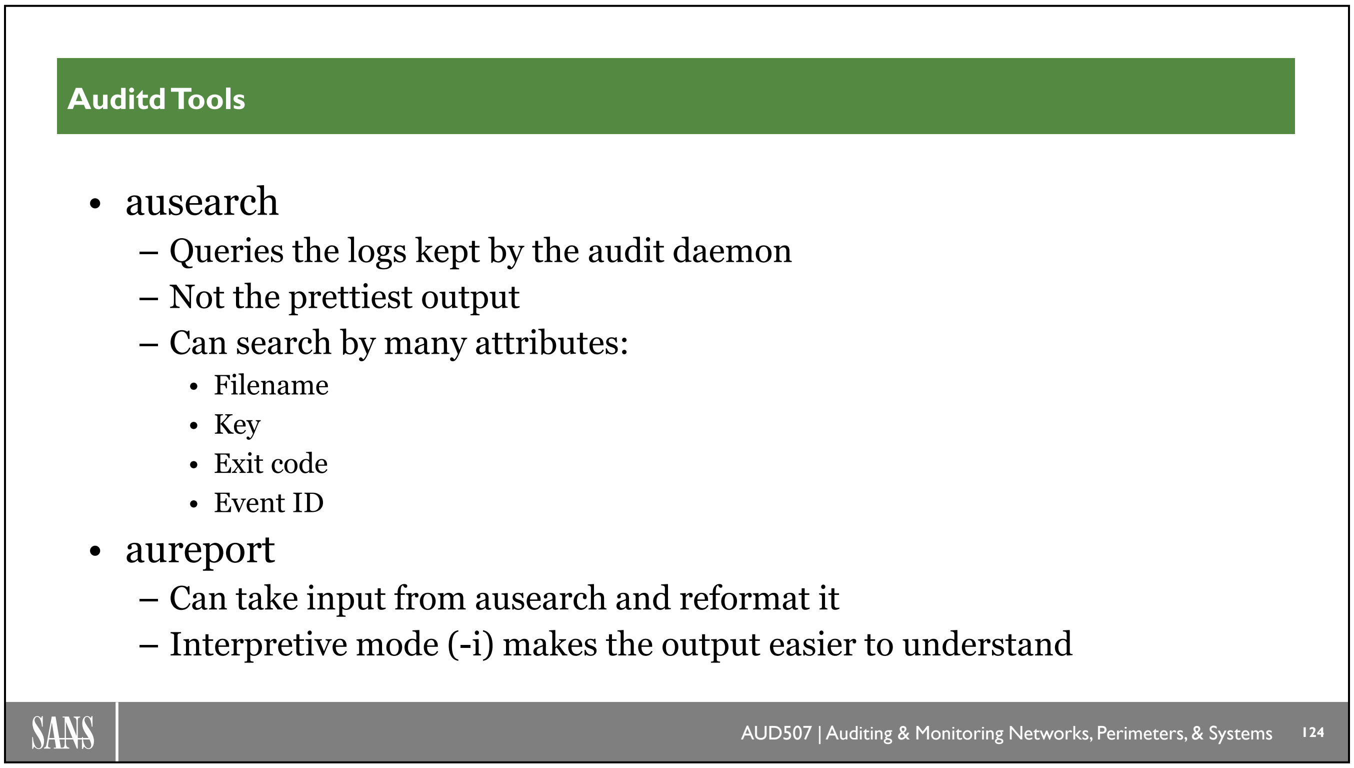
Even though this system creates text-based logs, there are two useful tools you might want to be aware of. Ausearch is a tool that allows for fast and efficient searching through these files, permitting you to specify fields of interest. This can be faster than grepping the file since grep will typically not be field-aware.

The other tool of interest is aureport, which allows you to generate statistics and other reports based on the audit logs. While all of this could be done with awk as well, using ausearch means that there is no need to write a complicated awk script to parse the files.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Auditd Tools

- ausearch
  - Queries the logs kept by the audit daemon
  - Not the prettiest output
  - Can search by many attributes:
    - Filename
    - Key
    - Exit code
    - Event ID
- aureport
  - Can take input from ausearch and reformat it
  - Interpretive mode (-i) makes the output easier to understand

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 124

The logs kept by the audit daemon can be searched using the powerful ausearch tool. It is capable of finding log entries based on many different criteria. Its output is not really intended for human consumption though, as it is verse terse.

To get a more readable report, pipe the output from ausearch into aureport with the -i option, which interprets numeric entries into readable names.

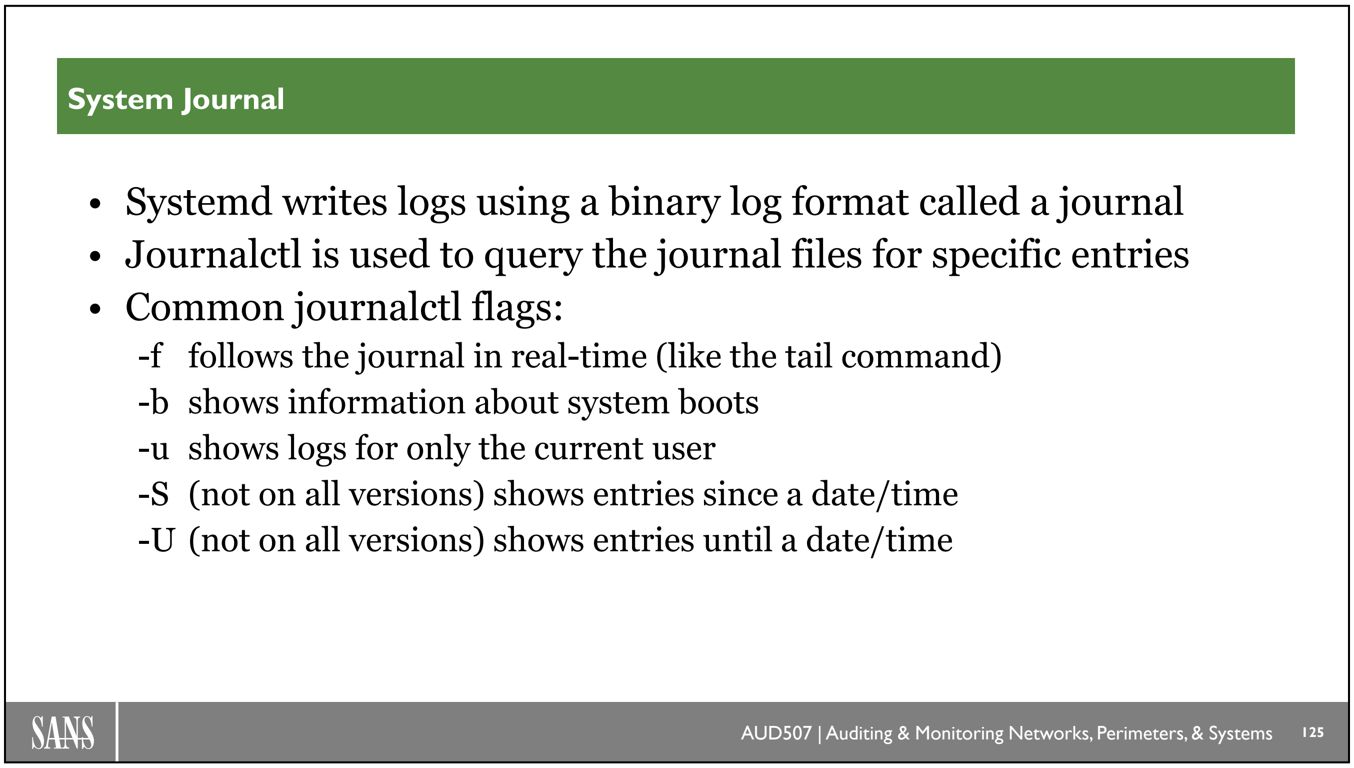
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



System Journal

- Systemd writes logs using a binary log format called a journal
- Journalctl is used to query the journal files for specific entries
- Common journalctl flags:

-f	follows the journal in real-time (like the tail command)
-b	shows information about system boots
-u	shows logs for only the current user
-S	(not on all versions) shows entries since a date/time
-U	(not on all versions) shows entries until a date/time

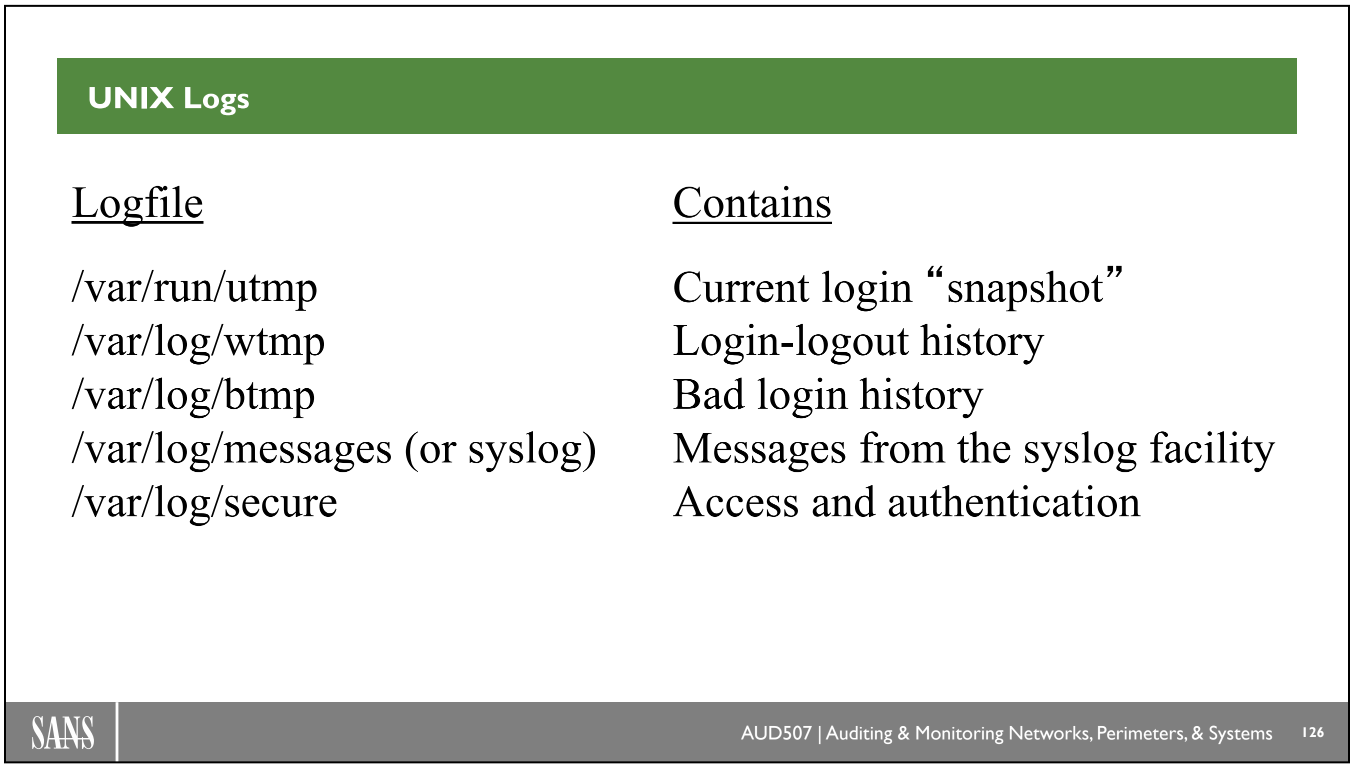
The system daemon, systemd, writes logs using binary log files on disk, known as journals. These files are normally stored in either the /var/log/journal or /run/log/journal directory. Since these files are binary, they cannot be read directly like the syslog or messages files. To read the information in the journals, you will use the journalctl tool.

The implementation details will vary by distribution, but journalctl usually offers many different ways to easily query the journals for the required data.

August 10, 2021

August 10, 2021

© SANS Institute 2021



UNIX Logs

Logfile	Contains
/var/run/utmp	Current login "snapshot"
/var/log/wtmp	Login-logout history
/var/log/btmp	Bad login history
/var/log/messages (or syslog)	Messages from the syslog facility
/var/log/secure	Access and authentication

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 126

The various log files of the UNIX operating system are often referred to as the "audit logs." So, it's no wonder that as you poke around on the host in question, you'll want to check that all of the "important events" are logged to the appropriate places. As you'll see, sometimes that involves making sure that the appropriate places exist in the first place. The following window shows a listing of the log files that are written in the /var/log directory. The auditor should examine each of these files because they should all have their own story to tell.

There are five major log files that the administrator should keep a watchful eye on for signs of unauthorized activity. For the most part, these files can be found in the /var/log directory of the Linux filesystem. The notable exception is utmp, which can be found in /var/run.

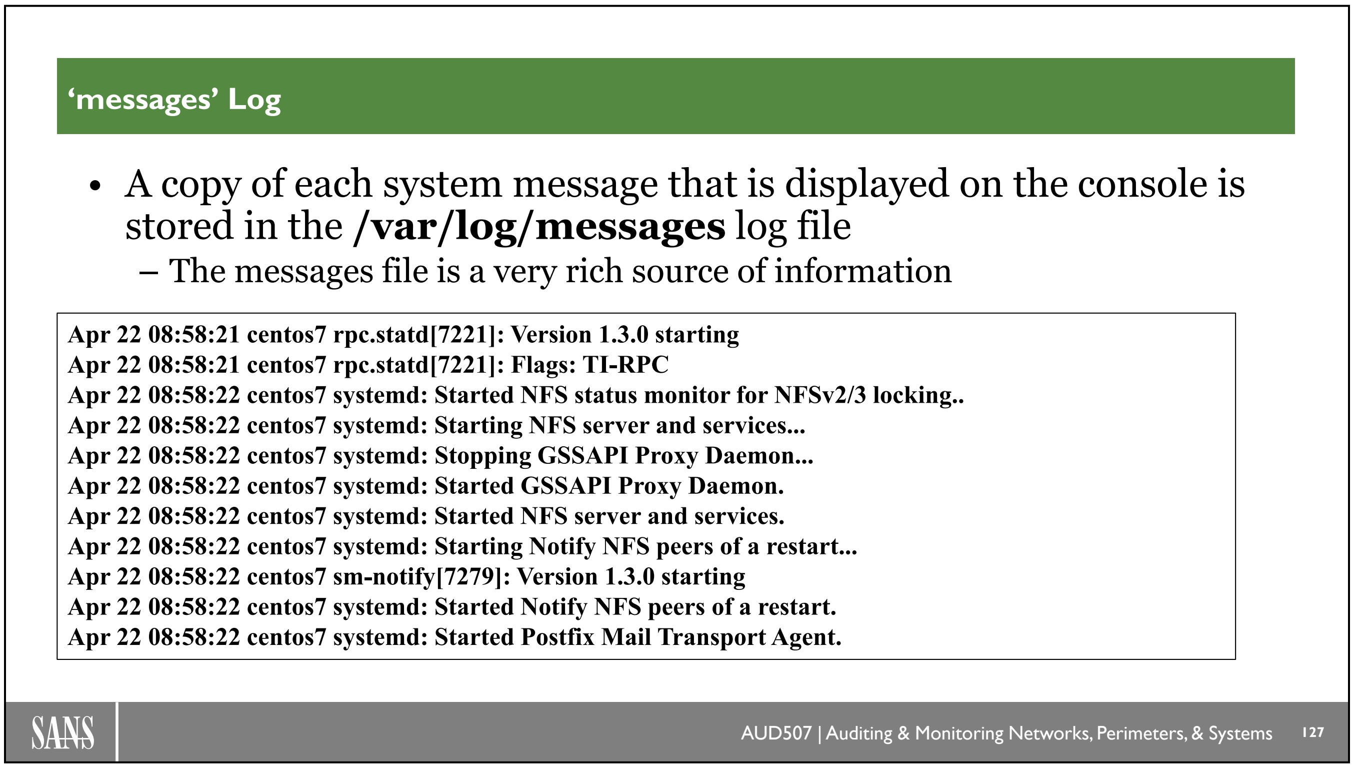
It is important to understand the contents of each of these log files and how one can exploit the information to achieve a more secure system. Starting with the next slide we discuss the details of each.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



‘messages’ Log

- A copy of each system message that is displayed on the console is stored in the **/var/log/messages** log file
  - The messages file is a very rich source of information

```
Apr 22 08:58:21 centos7 rpc.statd[7221]: Version 1.3.0 starting
Apr 22 08:58:21 centos7 rpc.statd[7221]: Flags: TI-RPC
Apr 22 08:58:22 centos7 systemd: Started NFS status monitor for NFSv2/3 locking..
Apr 22 08:58:22 centos7 systemd: Starting NFS server and services...
Apr 22 08:58:22 centos7 systemd: Stopping GSSAPI Proxy Daemon...
Apr 22 08:58:22 centos7 systemd: Started GSSAPI Proxy Daemon.
Apr 22 08:58:22 centos7 systemd: Started NFS server and services.
Apr 22 08:58:22 centos7 systemd: Starting Notify NFS peers of a restart...
Apr 22 08:58:22 centos7 sm-notify[7279]: Version 1.3.0 starting
Apr 22 08:58:22 centos7 systemd: Started Notify NFS peers of a restart.
Apr 22 08:58:22 centos7 systemd: Started Postfix Mail Transport Agent.
```

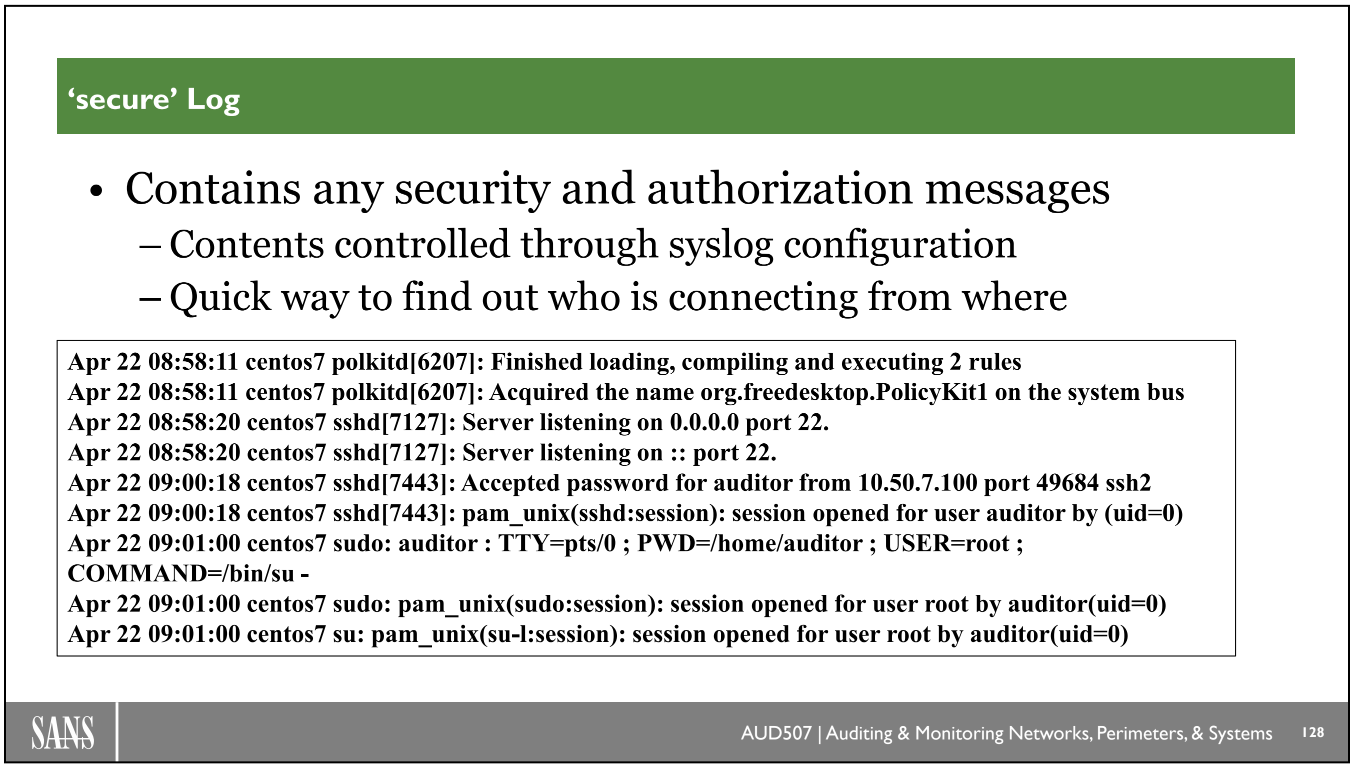
Each system message that gets sent to the console is also written to the messages file. Collectively, these messages are an incredibly rich source of information. Savvy system administrators can review the messages file to look for events that might indicate system trouble. Filled filesystems, failing devices, and system misconfigurations are just some of the tidbits that can be found in the messages file.

The ‘messages’ file is more traditionally known as the ‘syslog.’ Depending on how your system is configured, you might find that the messages file is actually named ‘syslog’! It is also possible that your system might have both a ‘syslog’ and a ‘messages’ file. In cases like this, it will be necessary to examine the ‘/etc/syslog.conf’ file, which is used to configure what and where things are logged, and to determine what sorts of information you can expect to find in each.

August 10, 2021

August 10, 2021

© SANS Institute 2021



‘secure’ Log

- Contains any security and authorization messages
  - Contents controlled through syslog configuration
  - Quick way to find out who is connecting from where

```
Apr 22 08:58:11 centos7 polkitd[6207]: Finished loading, compiling and executing 2 rules
Apr 22 08:58:11 centos7 polkitd[6207]: Acquired the name org.freedesktop.PolicyKit1 on the system bus
Apr 22 08:58:20 centos7 sshd[7127]: Server listening on 0.0.0.0 port 22.
Apr 22 08:58:20 centos7 sshd[7127]: Server listening on :: port 22.
Apr 22 09:00:18 centos7 sshd[7443]: Accepted password for auditor from 10.50.7.100 port 49684 ssh2
Apr 22 09:00:18 centos7 sshd[7443]: pam_unix(sshd:session): session opened for user auditor by (uid=0)
Apr 22 09:01:00 centos7 sudo: auditor : TTY=pts/0 ; PWD=/home/auditor ; USER=root ;
COMMAND=/bin/su -
Apr 22 09:01:00 centos7 sudo: pam_unix(sudo:session): session opened for user root by auditor(uid=0)
Apr 22 09:01:00 centos7 su: pam_unix(su-l:session): session opened for user root by auditor(uid=0)
```

The ‘secure’ log is used to record security and authentication messages on the system. It is quite likely that this information is also recorded in the ‘syslog’ or ‘messages’ file but creating a second copy in this log makes it easily accessible with minimal effort. The typical applications that will send messages to this file are the TCP Wrappers program, which is used for access control to the system, the PAM (Pluggable Authentication Modules) system, and the ‘login’ facilities on the system.

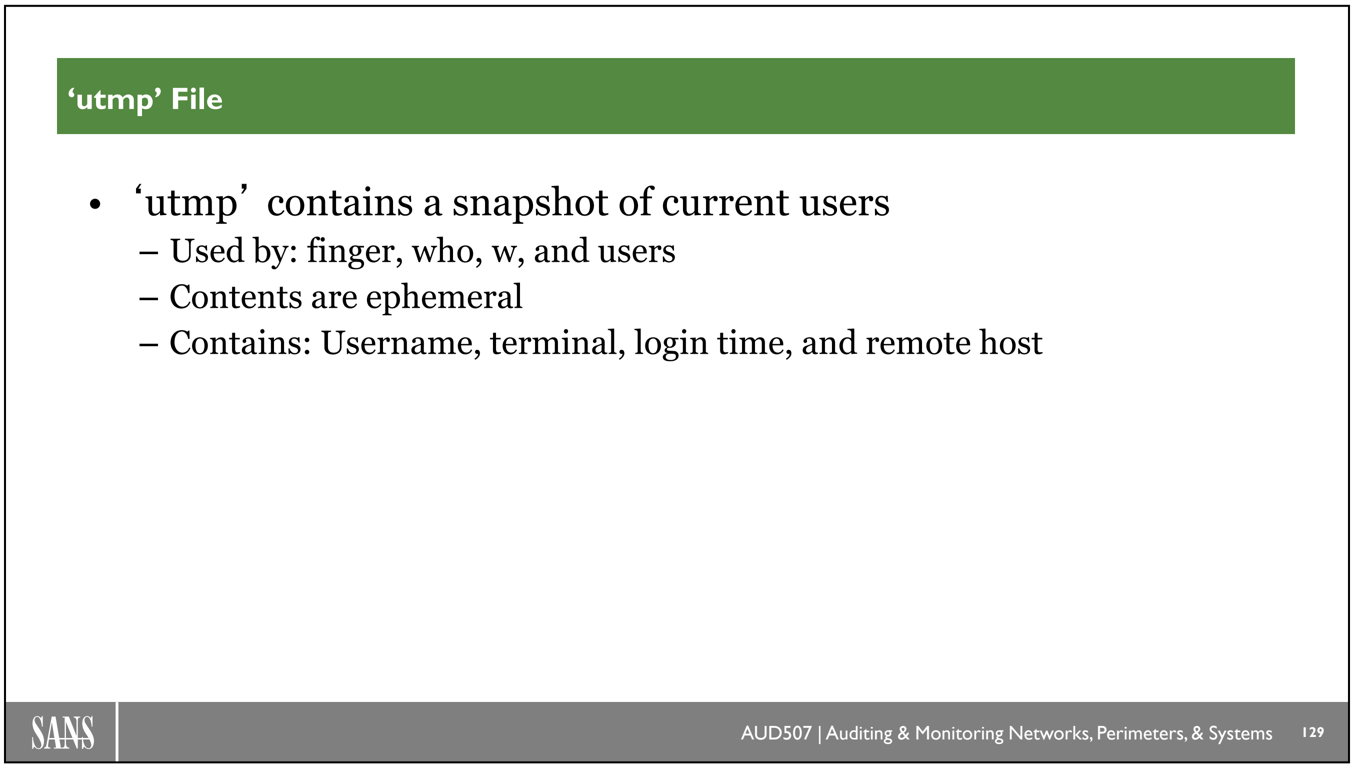
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



'utmp' File

- 'utmp' contains a snapshot of current users
  - Used by: finger, who, w, and users
  - Contents are ephemeral
  - Contains: Username, terminal, login time, and remote host

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 129

I used to work in a secure facility. Our employee badges had our identity and other pertinent information encoded on a strip on the back of the badge. People authorized to enter a building could do so by swiping their badge through a badge reader. This would record information and unlock the door. The exit process was the same. At any given time, physical security personnel could call up a list of people currently in the building.

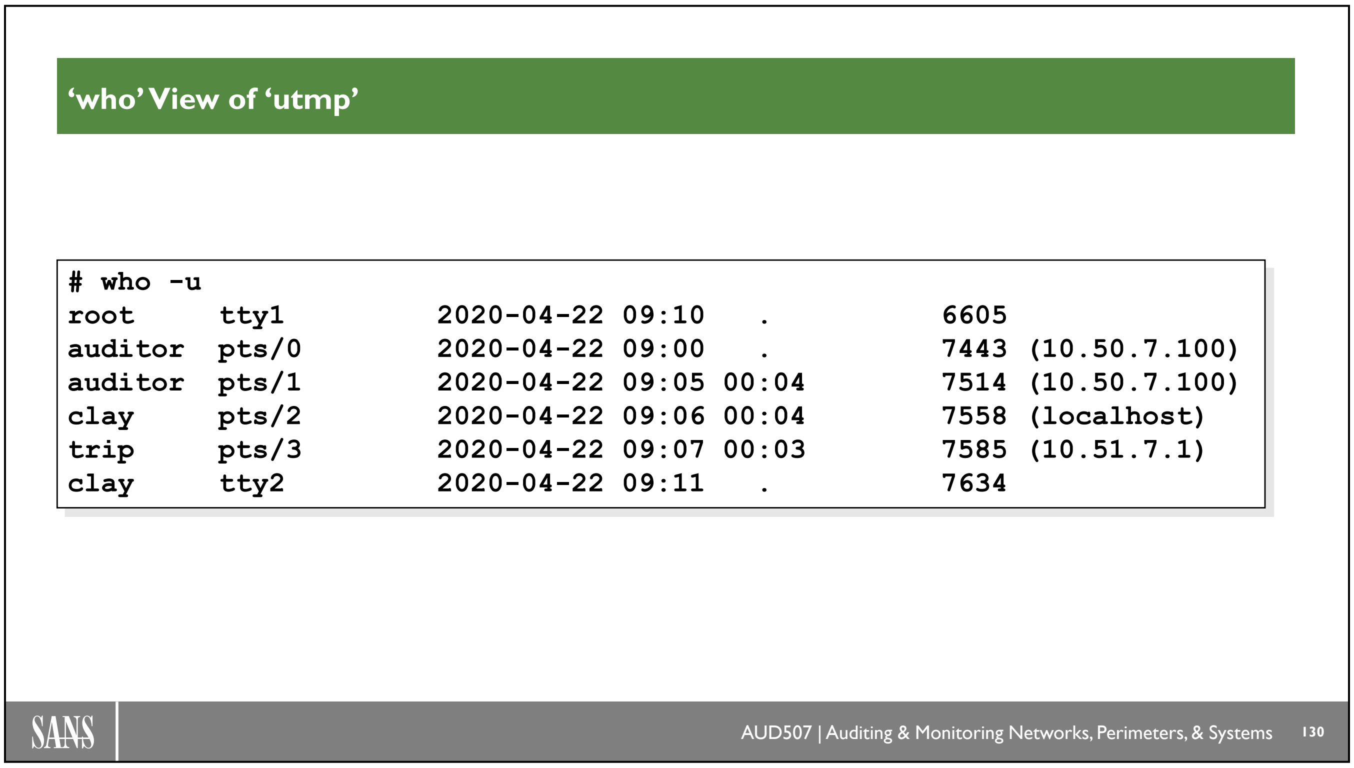
The utmp file works in a similar fashion. As users log in, an entry for them is made in the /var/run/utmp file. Upon logout, that entry is removed. The result is a file that contains a current user snapshot at any given point in time. As such, the contents of this file are short-lived and always changing.

For every successful login, utmp records, among other things, the username, device name, time, and origin. Unfortunately, utmp is a binary file, so it cannot be viewed with a text editor or other ASCII-based tool. However, programs like who, users, and finger read the utmp file and display its contents.

August 10, 2021

August 10, 2021

© SANS Institute 2021



‘who’ View of ‘utmp’

```
# who -u
root     tty1         2020-04-22 09:10   .          6605
auditor  pts/0        2020-04-22 09:00   .          7443 (10.50.7.100)
auditor  pts/1        2020-04-22 09:05 00:04        7514 (10.50.7.100)
clay     pts/2        2020-04-22 09:06 00:04        7558 (localhost)
trip     pts/3        2020-04-22 09:07 00:03        7585 (10.51.7.1)
clay     tty2         2020-04-22 09:11   .          7634
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 130

Here, we see the usage of the who command. The -u option has been given to show the users who are logged into the machine.

Root is apparently logged into the console of the machine (tty1), as is “clay” (tty2).

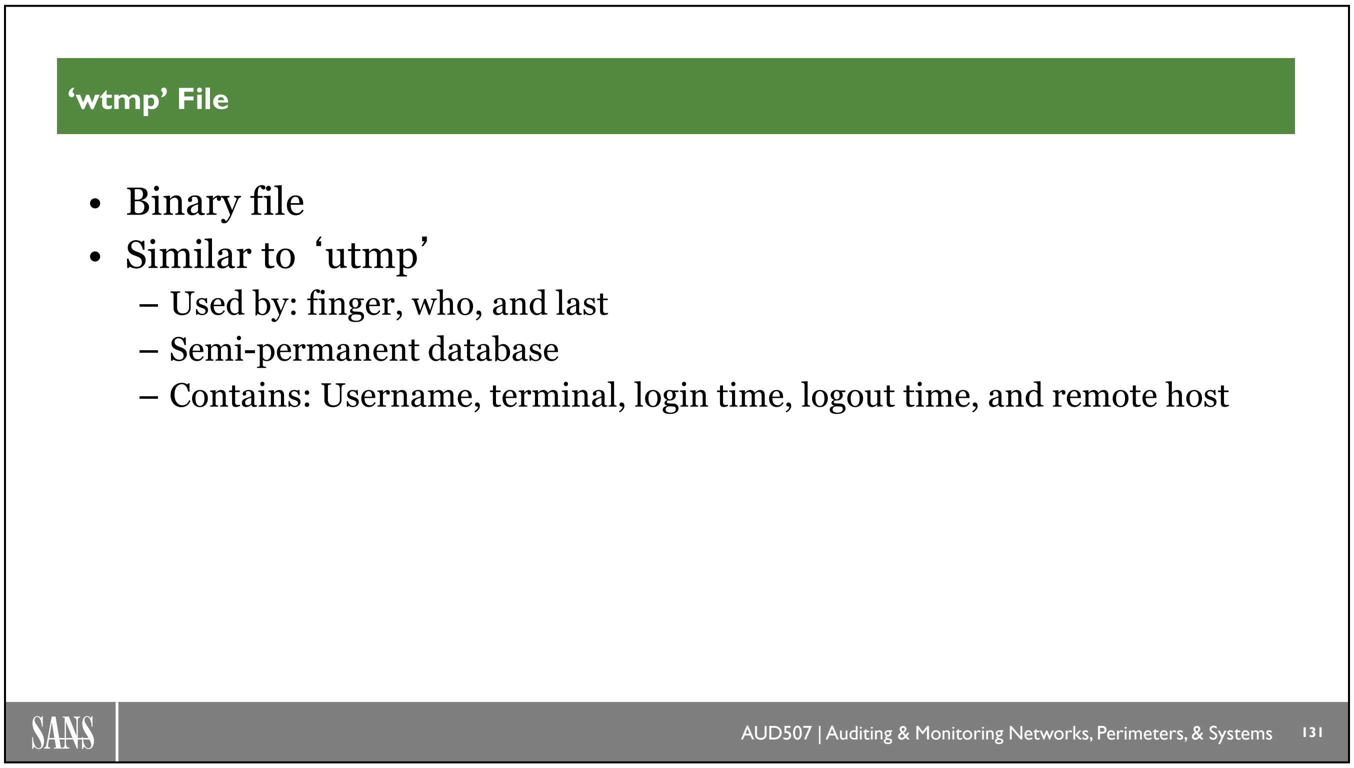
Auditor, trip, and clay are logged in via network connections. The client IP addresses are shown in the last column. Some of these sessions are “idle”—they have not been used in a few minutes (“00:03”).

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



'wtmp' File

- Binary file
- Similar to 'utmp'
  - Used by: finger, who, and last
  - Semi-permanent database
  - Contains: Username, terminal, login time, logout time, and remote host

SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 131

wtmp can be found in /var/log and is the same as utmp in terms of file type and format. It records the username, device, event time, and connection origin as a binary file. The major difference in file content lies in the fact that wtmp keeps a history of all logins, logouts, and system events. This provides a formal audit trail of user account access and host booting. Instead of being a user snapshot, wtmp is a running account and system history. This information is critical to intrusion detection and incident investigation efforts.

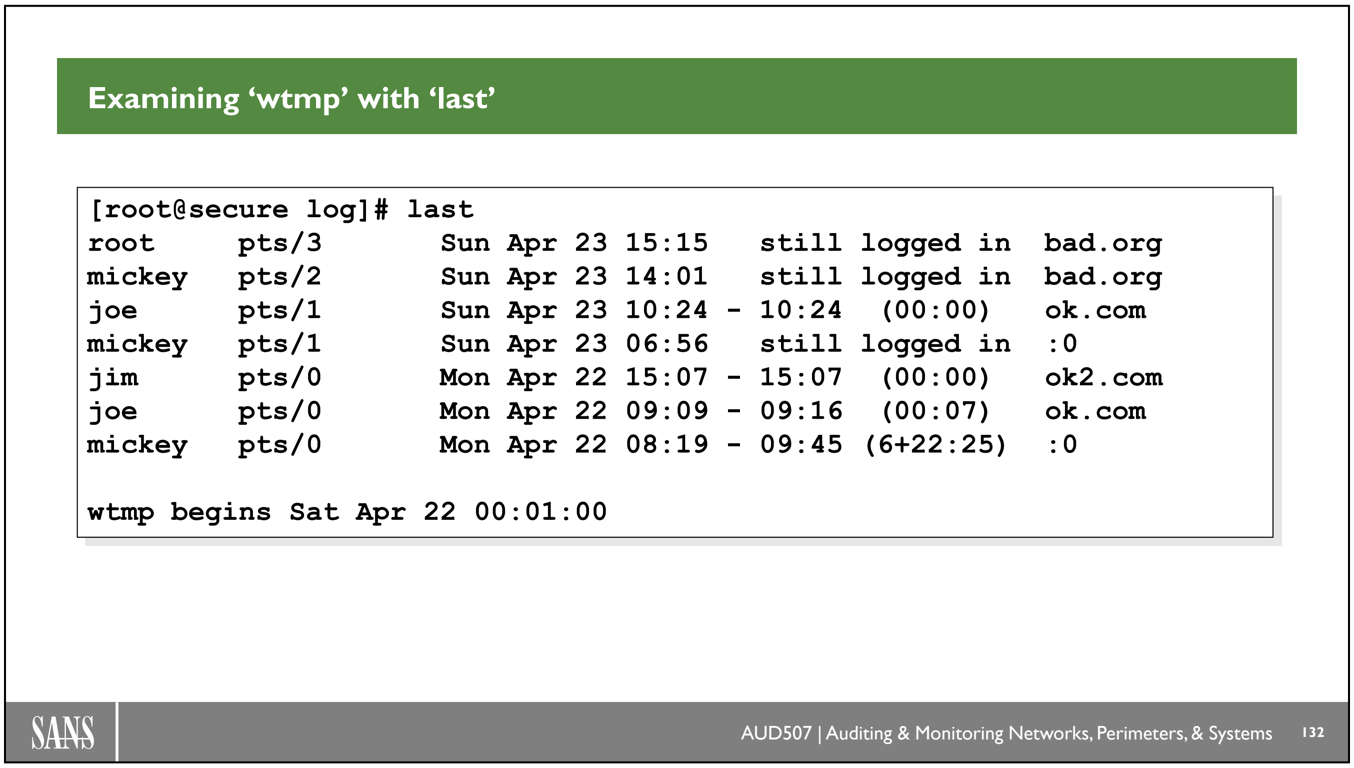
The contents of wtmp can be displayed with the who command, but the last command provides much more information. Administrators can prune the results of the who command to display only the last N events, or to show the account events for a particular user.

Typically, this file will persist for some number of days, weeks, or months. At the end of this period, the file will be automatically rotated. This means that the old file is archived, and a new file is created. The archived file will be retained along with a specified number of previous copies, all of which are configured using the logrotate system. We'll talk more about this at the end of this section.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Examining ‘wtmp’ with ‘last’

```
[root@secure log]# last
root     pts/3       Sun Apr 23 15:15   still logged in   bad.org
mickey   pts/2       Sun Apr 23 14:01   still logged in   bad.org
joe      pts/1       Sun Apr 23 10:24 - 10:24  (00:00)    ok.com
mickey   pts/1       Sun Apr 23 06:56   still logged in   :0
jim      pts/0       Mon Apr 22 15:07 - 15:07  (00:00)    ok2.com
joe      pts/0       Mon Apr 22 09:09 - 09:16  (00:07)    ok.com
mickey   pts/0       Mon Apr 22 08:19 - 09:45 (6+22:25)   :0

wtmp begins Sat Apr 22 00:01:00
```

When using ‘last’, the first thing to be aware of is that it displays the contents of the wtmp file in a “most recent to least recent” time order. Said differently, entries at the top of the output occurred most recently. The beginning date of the file is on the very last line. This wtmp began on April 22nd at 1 minute after midnight.

Scanning left to right, we see the username, the tty, the login and logout time, and the host of origin. Elapsed activity time is displayed in parentheses to the right of event times. Also, for those users still logged in, a corresponding message is displayed in place of the logout time.

If we examine the output from the bottom up, we can read a history of the activity on the host. The entries look fairly normal until mickey’s entry on April 23 at 14:01. We can see that mickey logged in from the console (the ":0" entry) at 6:56 on April 23rd and was still logged in when the last command was run. The frightening lines are the first two. Apparently, someone is using mickey’s account from the host bad.org. In this example, someone also acquired a root shell about an hour later from that same host. Could a sniffed password have been the vehicle for a local compromise?

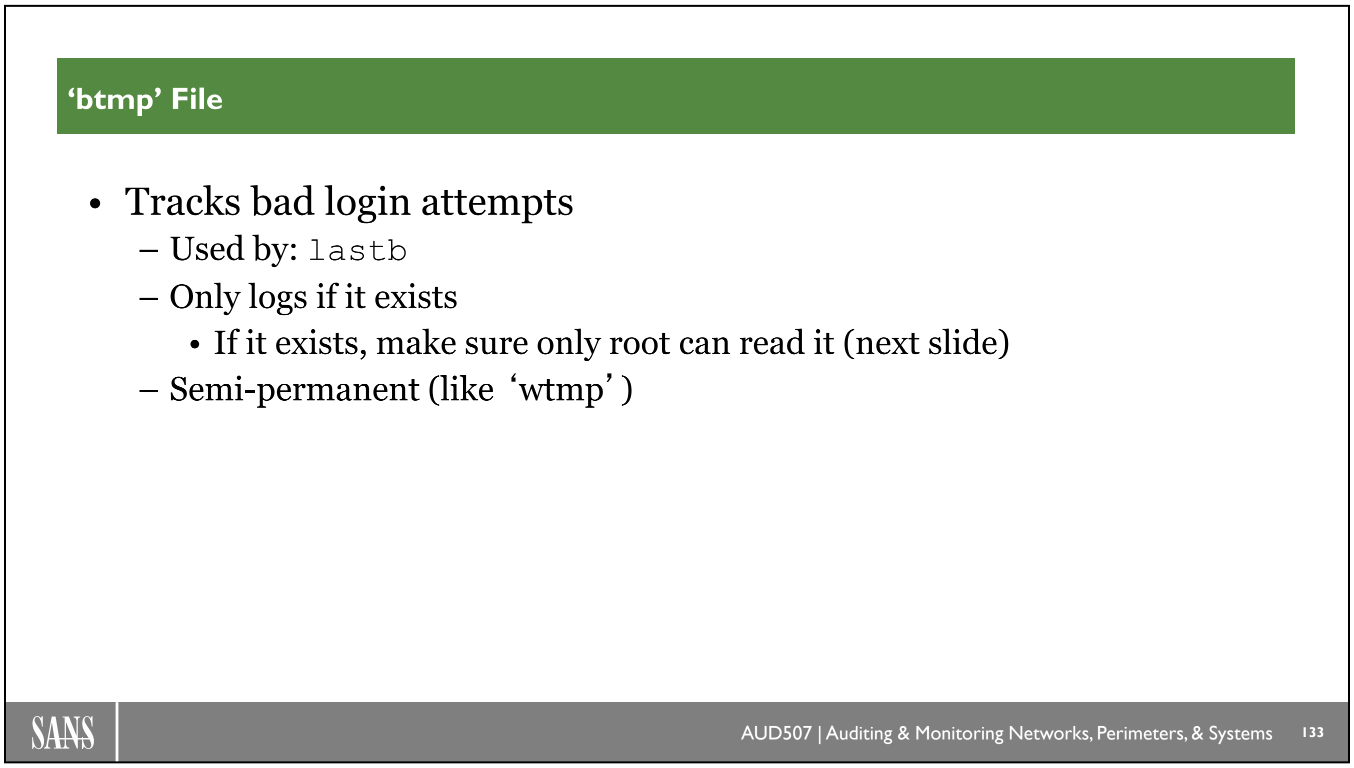
As you can see, the wtmp file is a necessity for reconstructing system events in an investigation. But be warned! wtmp must be activated, or no such logs are collected.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



'btmp' File

- Tracks bad login attempts
  - Used by: `lastb`
  - Only logs if it exists
    - If it exists, make sure only root can read it (next slide)
  - Semi-permanent (like 'wtmp')

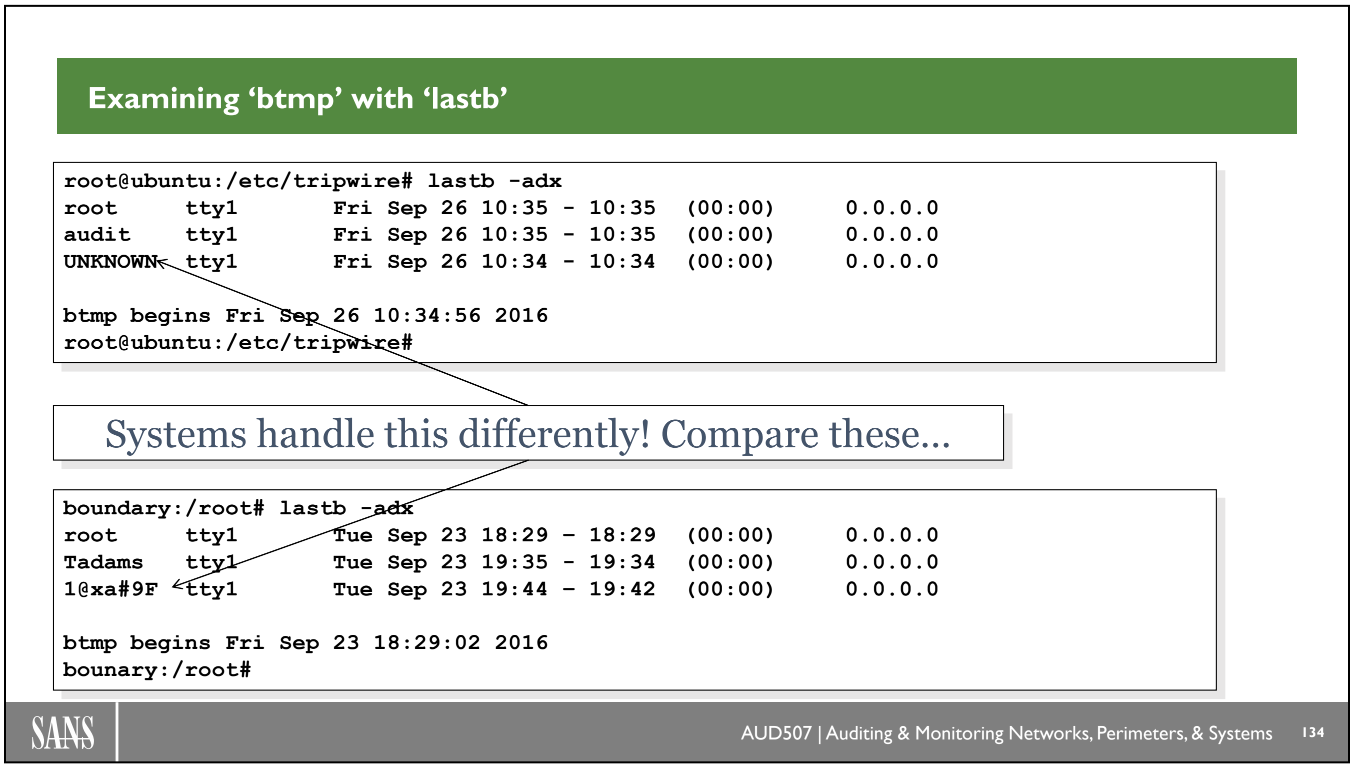
/var/log/btmp is the file used to log bad login attempts. It records the username, the device, the time, and the origin of the failed login attempt. The lastb command can be used to examine its contents because, like the others, it is a binary file.

Like wtmp, the btmp file must exist in the /var/log/ directory in order for bad logins to be recorded. We can use the ls command to check for the file's existence. If it doesn't exist, create btmp using the touch command as described earlier. Make sure that btmp's permissions are set so that only root can read from it!

August 10, 2021

August 10, 2021

© SANS Institute 2021



Examining 'btmp' with 'lastb'

```
root@ubuntu:/etc/tripwire# lastb -adx
root     tty1       Fri Sep 26 10:35 - 10:35  (00:00)     0.0.0.0
audit    tty1       Fri Sep 26 10:35 - 10:35  (00:00)     0.0.0.0
UNKNOWN  tty1       Fri Sep 26 10:34 - 10:34  (00:00)     0.0.0.0

btmp begins Fri Sep 26 10:34:56 2016
root@ubuntu:/etc/tripwire#
```

Systems handle this differently! Compare these…

```
boundary:/root# lastb -adx
root     tty1       Tue Sep 23 18:29 - 18:29  (00:00)     0.0.0.0
Tadams   tty1       Tue Sep 23 19:35 - 19:34  (00:00)     0.0.0.0
1@xa#9F  tty1       Tue Sep 23 19:44 - 19:42  (00:00)     0.0.0.0

btmp begins Fri Sep 23 18:29:02 2016
bounary:/root#
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 134

Here, we can see two example outputs from two different UNIX systems. In the first case, we can see the logon failures for a user named "audit," a user named "root," and an UNKNOWN user. Why UNKNOWN? The user entered a value that was not recognized as a UNIX system. This, as it turns out, is a very important protection! Unfortunately, not all UNIX systems have this behavior.

Consider the output in the lower example. Here, we see a system named "Boundary" reporting logon failures for "root," "Tadams," and "1@xa#9F." Look at that last username… Does that look sort of suspicious? In fact, it doesn't look anything like a username. What's happening here?

In this case, the btmp is not masking unknown entries. What most likely happened is that a user just accidentally typed his password into the username prompt. Without the masking feature, we end up potentially recording passwords!

Remember that we said that only the root user should be able to read this file? This is why. Enterprising users can simply troll this file (if it's readable) for things that look like passwords. When they find one, they can simply look at the output of 'last' to see who logged in right *after* the password hit the btmp file.

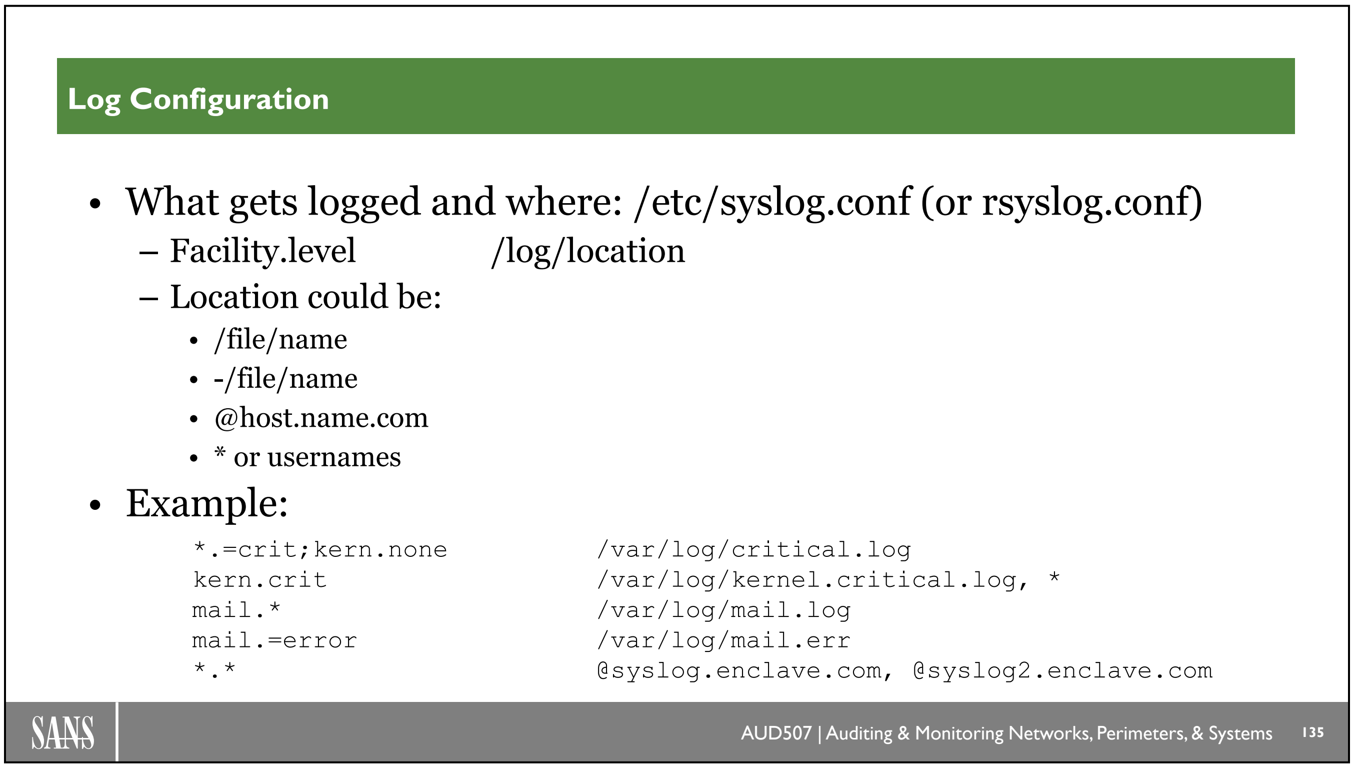
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Log Configuration

- What gets logged and where: /etc/syslog.conf (or rsyslog.conf)
  - Facility.level /log/location
  - Location could be:
    - /file/name
    - -/file/name
    - @host.name.com
    - * or usernames
- Example:

```
*.=crit;kern.none          /var/log/critical.log
kern.crit                  /var/log/kernel.critical.log, *
mail.*                     /var/log/mail.log
mail.=error                /var/log/mail.err
*.*                        @syslog.enclave.com, @syslog2.enclave.com
```

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 135

Configuration of the logging is largely controlled through two files. The first is syslog.conf. This is the main logging configuration that defines what gets logged where.

Each configuration line in the file will start with a facility description (kernel, daemon, etc.) followed by a logging level (info, debug, emergency, etc.). These can be lists of facilities and levels or even wildcards. This "selector," if you will, is used to define what will be put into the entry on the right-hand side of the line. To the right, you will typically find a log path and filename. If that file path is prefixed with a '-', the log will not be "synced"' after every write. What this means is that it will cache log entries and write them in batches. For critical events, you would prefer that the log is synced immediately, though this creates more load on the system.

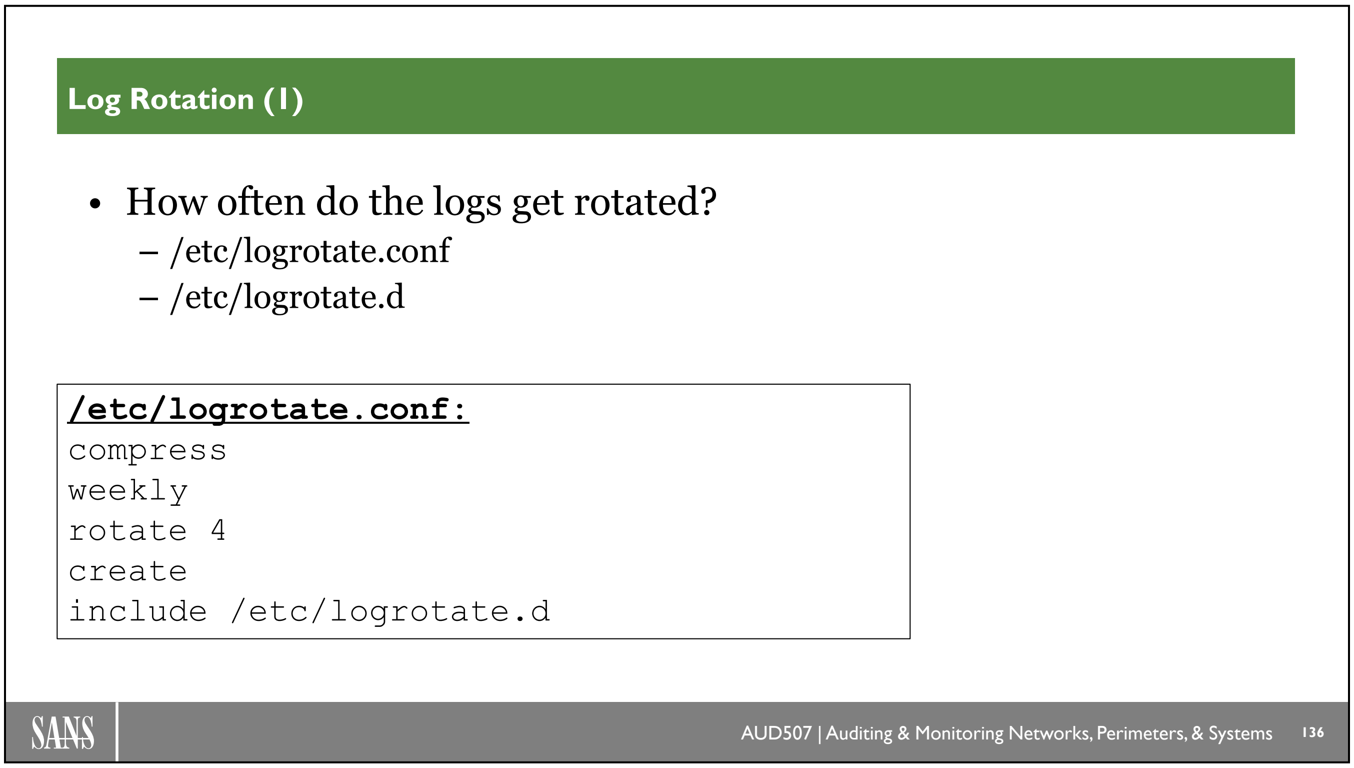
You might also find an @ followed by an IP address or hostname, automatically configuring the system to log remotely. Finally, you could find a username, a list of usernames, or even an asterisk. This will cause the matching log messages to be broadcast to the users listed if they are logged in or—in the case of the asterisk, to all logged in users.

The rotation of logs is controlled by logrotate these days. The primary configuration governing how often to rotate logs and how many copies to keep is found in the logrotate.conf file. Individual configurations for various services can be found in the logrotate.d directory.

Considering the examples in the slide, you can see that the first line will record all events that are marked critical, and only critical, but not if they are coming from the kernel. The next line creates a separate line for kernel critical messages, which will be directed to a file and which will alert all logged-on users (the * indicates this). Next, we have a catch-all for mail logs, followed by an error-specific log file. Finally, all messages are forwarded to two central log servers.

August 10, 2021

© SANS Institute 2021



Log Rotation (1)

- How often do the logs get rotated?
  - /etc/logrotate.conf
  - /etc/logrotate.d

```
/etc/logrotate.conf:
compress
weekly
rotate 4
create
include /etc/logrotate.d
```

SANS | AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 136

The logrotate system within UNIX is quite useful. It is structured similarly to the xinetd system in that there is a global, top-level configuration file and a directory within which service-specific configurations are found.

The top-level configuration will specify global options for all logs. These settings can be overridden within the individual log configuration files. In the case that we have in the slide, logs will rotate weekly, by default, and the system will retain the last four rotate versions of a log. Additionally, the logs will be compressed when they are rotated, saving space on the log drive. The 'create' option indicates that a new file will be created immediately after the old log is rotated. It is possible to specify file ownership and permissions with the create option, but that was not done here.

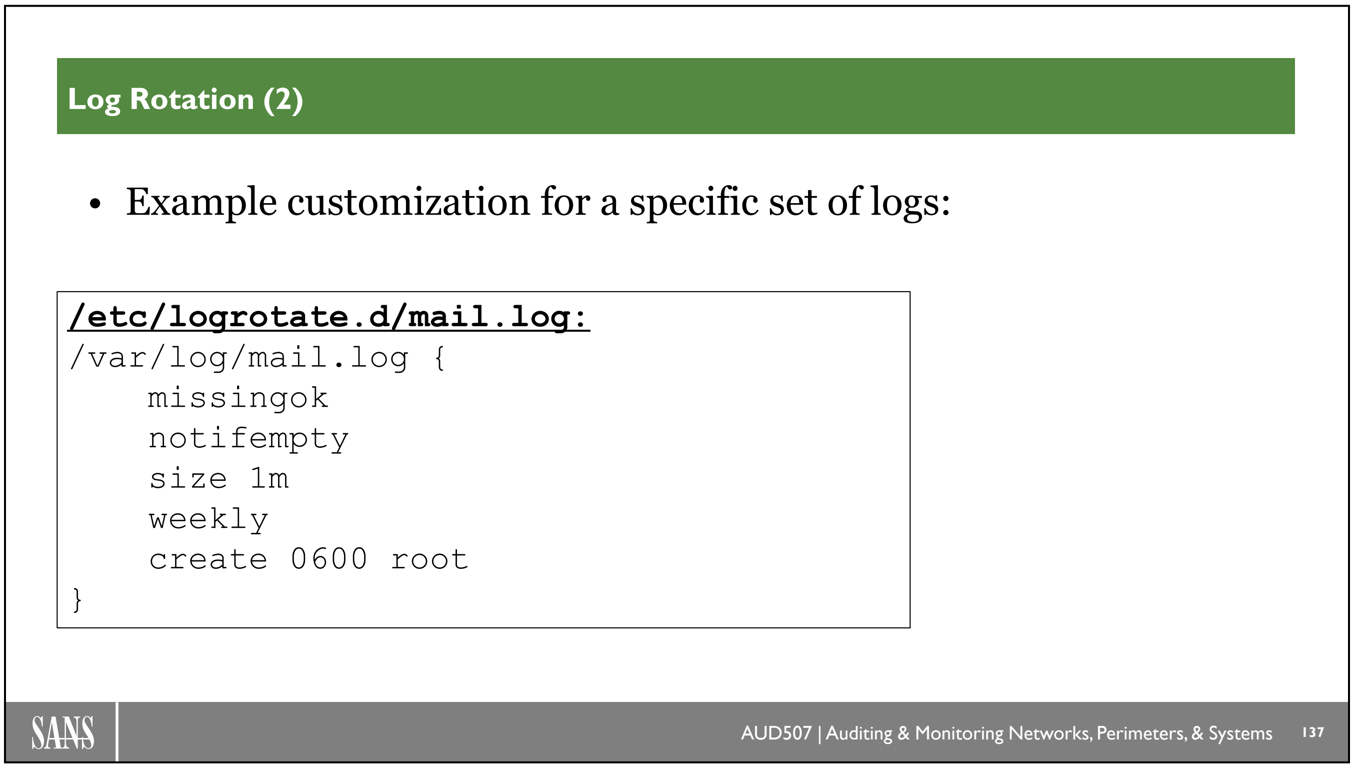
Finally, the last line in the configuration tells logrotate to load all of the configuration files within the /etc/logrotate.d directory.

August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021



Log Rotation (2)

- Example customization for a specific set of logs:

```
/etc/logrotate.d/mail.log:
/var/log/mail.log {
    missingok
    notifempty
    size 1m
    weekly
    create 0600 root
}
```

More granular log settings can be configured in files located within the /etc/logrotate.d directory. In this case, we are looking at the configuration for /var/log/mail.log. Here's how this configuration breaks down.

First, we indicate that it is okay if the log file doesn't exist. If it doesn't exist, the system will simply move on. Without this option, or with nomissingok configured, logrotate will generate an error if the log file is missing. This might be a good thing to do to identify mysteriously deleted log files on our systems.

The next line, notifempty, tells logrotate that if the log is empty, there is no need to rotate it. This is actually quite a useful option. Imagine we had our logs configured to rotate daily. Further, imagine that there is a log that has very little activity in it over time. If we keep seven copies and rotate daily, we might have seven logs that are completely empty. With this option, the log will only rotate if the file has content; therefore, quiet logs will only rotate when there is content, allowing us to look back and see the last time anything hit this log, even if it would typically be outside of the log rotation window.

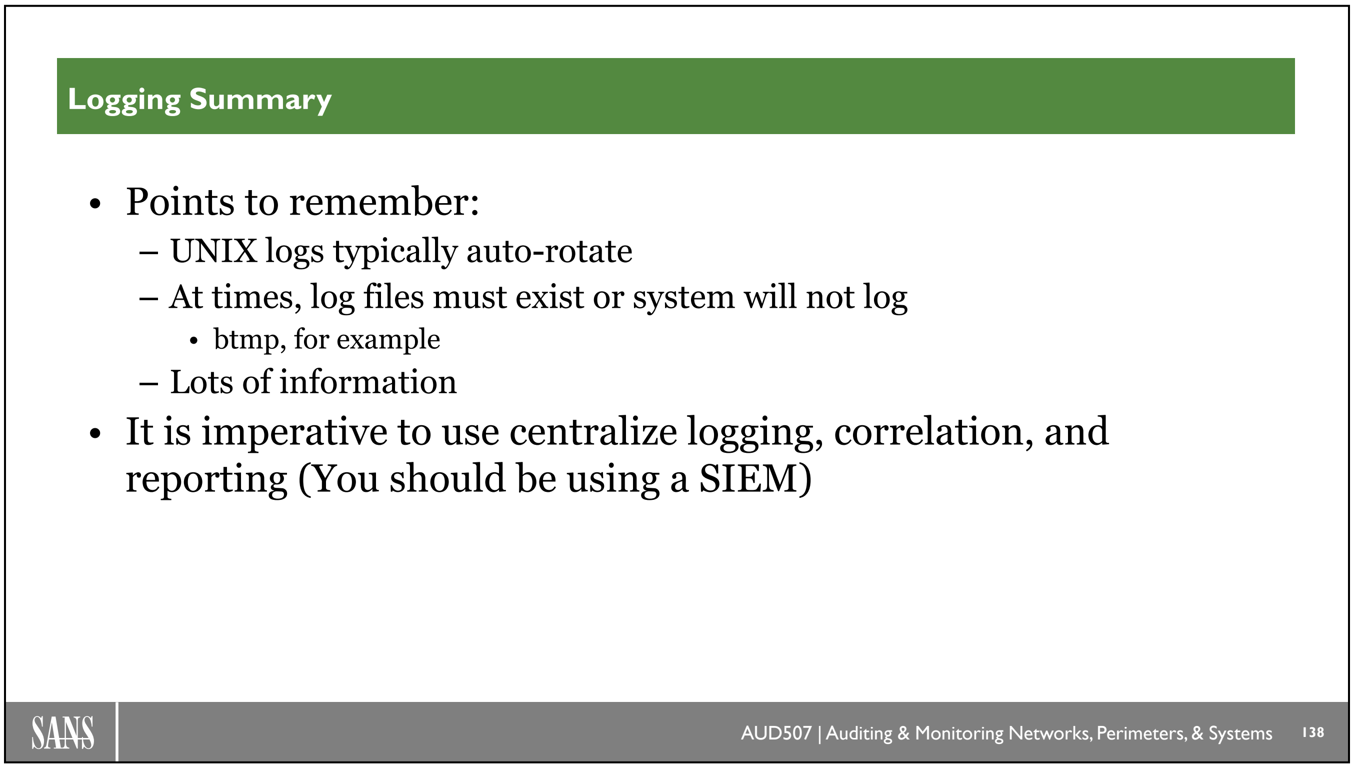
The next option allows us to specify the maximum size that the log will be permitted to grow to. Additionally, the line after that specifies weekly. These two options work together. If the log file grows to be greater than 1 megabyte in less than a week, it will be rotated. However, if a week goes by and the file still isn't 1 megabyte, it will be rotated at the week mark.

Finally, we see the create option again. Here, create is being told to set the file permissions to read-write, and the ownership is being changed to root:root.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Logging Summary

- Points to remember:
  - UNIX logs typically auto-rotate
  - At times, log files must exist or system will not log
    - btmp, for example
  - Lots of information
- It is imperative to use centralize logging, correlation, and reporting (You should be using a SIEM)

Just a few points to remember so far: UNIX log files are typically configured to automatically rotate periodically; in order for a log entry to be recorded, the log file must exist (UNIX tends to believe that if the log file doesn't exist, you must not want to know); and there's a great deal of information to be found in these logs. We'll examine a variety of ways to trim down the volume of these logs, focusing on the most interesting bits of information.

As we have mentioned throughout the week, it is very important to centralize the collection of logs into some sort of a security information and event management (SIEM) system. With the advent of virtualization, most enterprises have too many servers to be able to manage and examine logs on individual systems.

August 10, 2021

© 2021 Risenhoover Consulting, Inc.

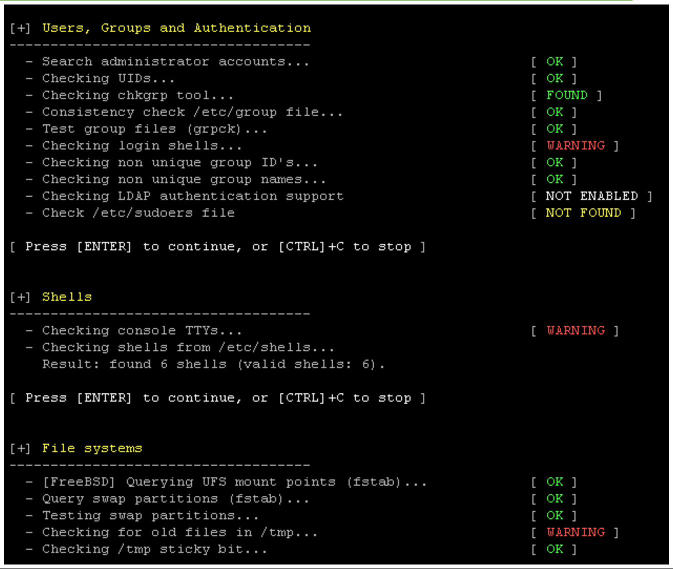
August 10, 2021

Technet24

© SANS Institute 2021

Host-Based Assessment

- Lynis is a configuration testing tool for many *nixes
  - Supports: AIX, FreeBSD, HP/UX, Linux, OS X, NetBSD, OpenBSD, Solaris
- Audits a system based on discovered services and configuration
- Open source with commercial enterprise edition



```
[+] Users, Groups and Authentication
------------------------------------
  - Search administrator accounts...                          [ OK ]
  - Checking UIDs...                                          [ OK ]
  - Checking chkgrp tool...                                   [ FOUND ]
  - Consistency check /etc/group file...                      [ OK ]
  - Test group files (grpck)...                               [ OK ]
  - Checking login shells...                                  [ WARNING ]
  - Checking non unique group ID's...                         [ OK ]
  - Checking non unique group names...                        [ OK ]
  - Checking LDAP authentication support                      [ NOT ENABLED ]
  - Check /etc/sudoers file                                   [ NOT FOUND ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] Shells
------------------------------------
  - Checking console TTYs...                                  [ WARNING ]
  - Checking shells from /etc/shells...
    Result: found 6 shells (valid shells: 6).

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] File systems
------------------------------------
  - [FreeBSD] Querying UFS mount points (fstab)...            [ OK ]
  - Query swap partitions (fstab)...                          [ OK ]
  - Testing swap partitions...                                [ OK ]
  - Checking for old files in /tmp...                         [ WARNING ]
  - Checking /tmp sticky bit...                               [ OK ]
```

Lynis is a host-based assessment tool for most modern Unixes. It performs a suite of configuration checks "opportunistically," by scanning whatever it finds running or configured on the system. We have found that we can get good reports from Lynis with very little configuration.

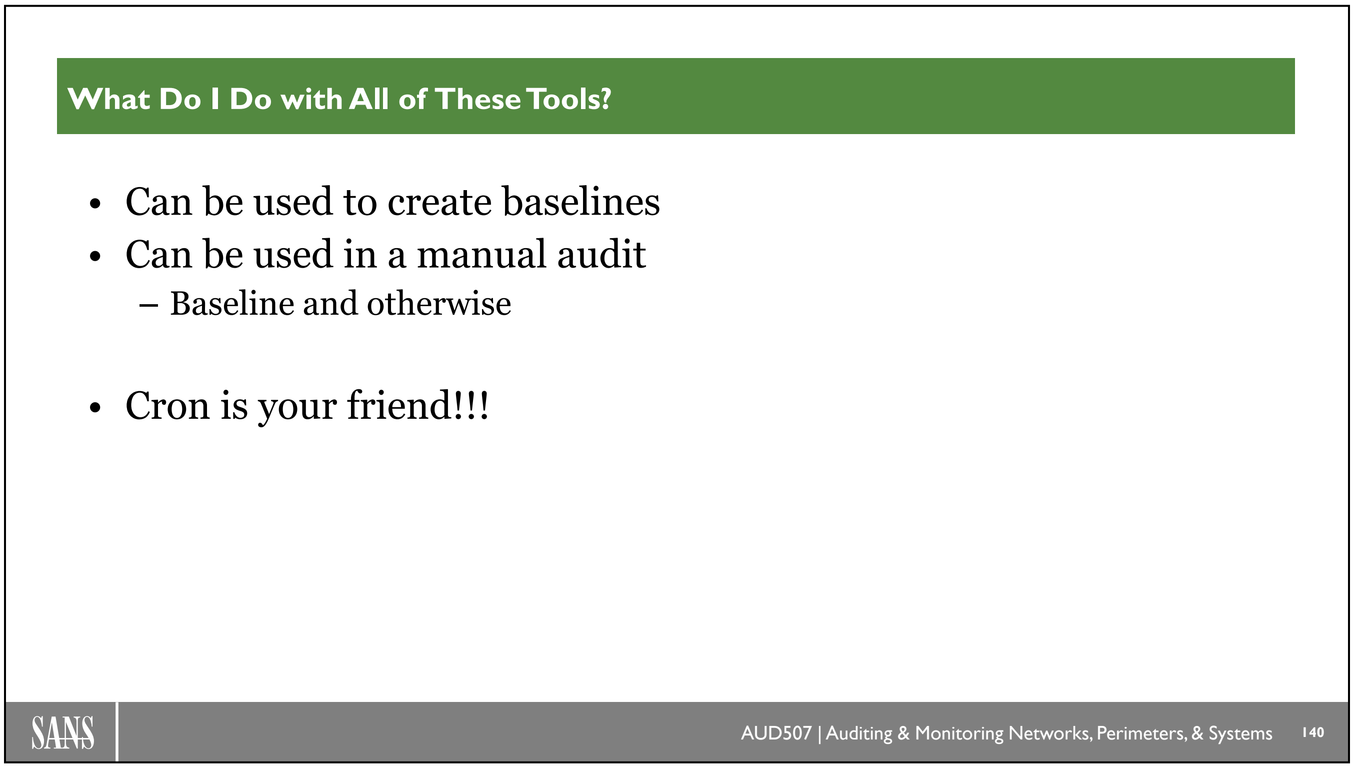
Lynis doesn't test against a particular standard, but there is good news. The tests for Lynis are written as shell scripts, and you can build your own to meet your organization's needs. You can also enable and disable tests to suit your environment.

The enterprise version of Lynis includes extra plugins and centralized control and monitoring of systems.

August 10, 2021

August 10, 2021

© SANS Institute 2021



What Do I Do with All of These Tools?

- Can be used to create baselines
- Can be used in a manual audit
  - Baseline and otherwise

- Cron is your friend!!!

Although all of the tools mentioned on the last few pages can be used to gather baseline information and can be used to perform a manual security audit where baselines don't exist, they are also prime for scripting! If we could script a few of these tools together and have the script run periodically, we could potentially create automated audit reports or exception reports with a minimal amount of effort.

When you begin to create automated reports, it is important to consider the impact that these can have on a system administrator or security officer. If we produce lots of noisy but essentially unimportant reports, the reports will begin to be ignored. On the other hand, if the reports that are created are easy to peruse and report only on exceptions rather than informational "noise," they can become excellent pieces in our overall Defense-in-Depth or Time-Based Security paradigm.

The UNIX facility that you will want to use to make this happen is 'cron.'

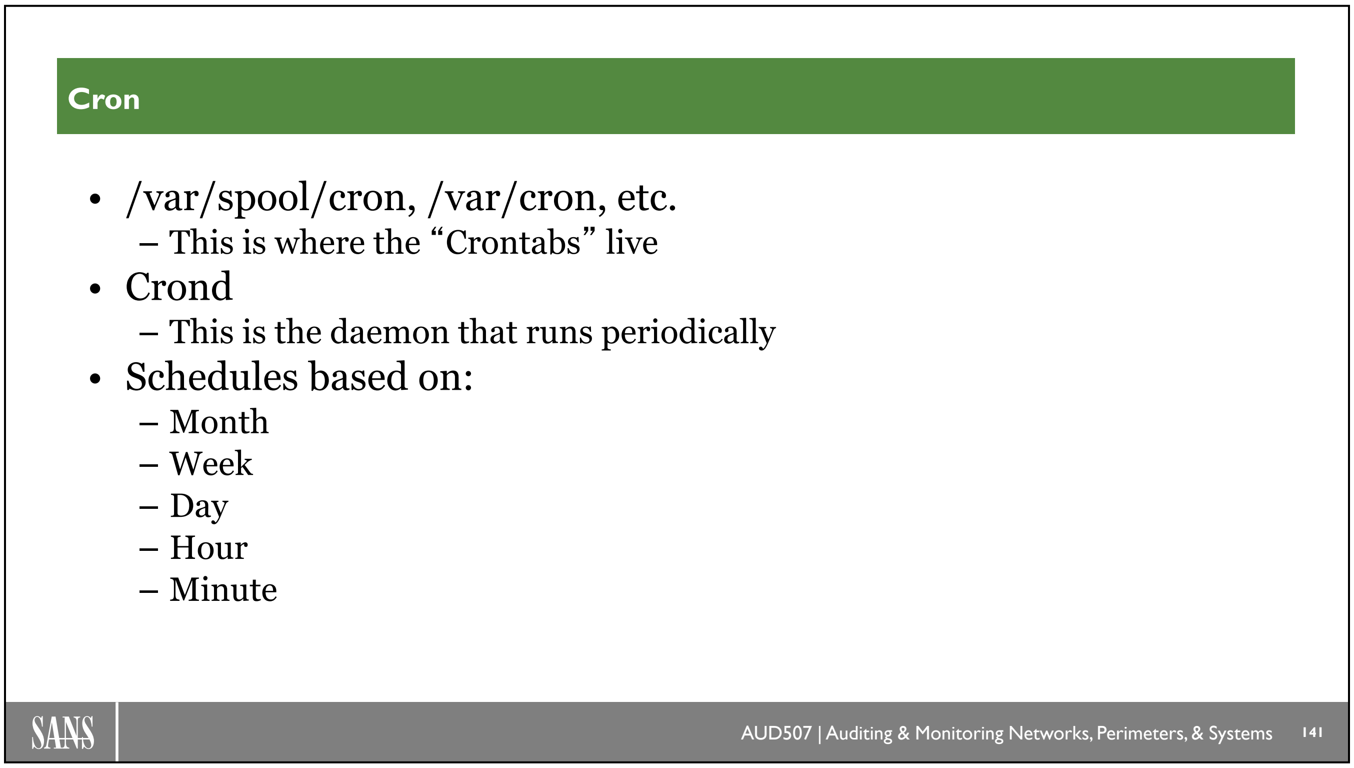
August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021



Cron

- /var/spool/cron, /var/cron, etc.
  - This is where the "Crontabs" live
- Crond
  - This is the daemon that runs periodically
- Schedules based on:
  - Month
  - Week
  - Day
  - Hour
  - Minute

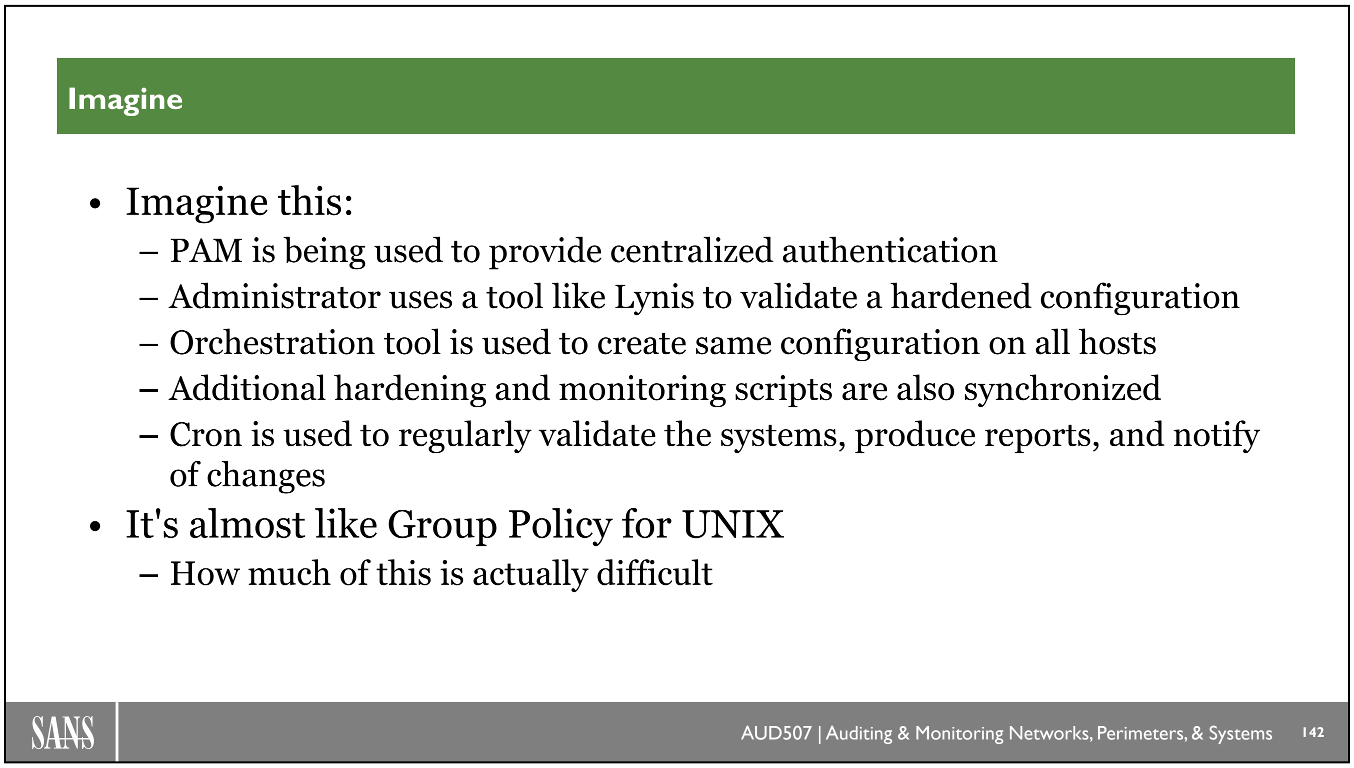
The cron daemon is a very flexible scheduling daemon that controls scheduled jobs for all users. Quite likely, even if your administrators don't think they are running cron jobs, there are system cron jobs that run periodically. Usually, UNIX systems will come with automated facilities for rotating logs and performing regular security checks on the system. These jobs are controlled by cron.

The cron daemon also has a great deal of granularity. Using cron, you can schedule jobs to run tomorrow, next week, next month, twice per month, twice per day, five times per hour, and any other combination that you can think of. This means that we could potentially configure a cron job to run all of our baseline audit tools periodically—say, every day or maybe even every hour—and report differences or exceptions to the system administrator or a security officer. The tools that would run are Tripwire, chkwtmp, raudit, and so on.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Imagine

- Imagine this:
  - PAM is being used to provide centralized authentication
  - Administrator uses a tool like Lynis to validate a hardened configuration
  - Orchestration tool is used to create same configuration on all hosts
  - Additional hardening and monitoring scripts are also synchronized
  - Cron is used to regularly validate the systems, produce reports, and notify of changes
- It's almost like Group Policy for UNIX
  - How much of this is actually difficult

Before we conclude, please take a moment to think about how all of this integrates. Imagine that we have an environment where all of the authentication to the UNIX systems is brokered by PAM, perhaps integrated into the Active Directory of our domain. Add to this the use of a tool like Lynis, which is especially interesting since it supports multiple flavors of UNIX. What if the administrator were to take the time to generate a common configuration that could be fed to Lynis to generate a compliance report? Could he even create more specific configurations that apply to unusual versions of UNIX that your organization might be running, and then deploy them using Chef or Puppet?

In fact, he could also push out other useful monitoring scripts and tools, perhaps ones that he has written or others like chkrootkit. What if he were to configure every system to periodically run these scripts, generate reports, send out alerts, or reapply settings? In a very real sense, it is as though we have created a Group Policy analogue!

One final question to think about. How much of what is described here is actually difficult? Certainly, aspects of it might be beyond the abilities of the average auditor, but if you were able to follow along and understand the discussion in this course, then you can see what's involved in this process. Should an administrator be able to implement such controls? Are they?

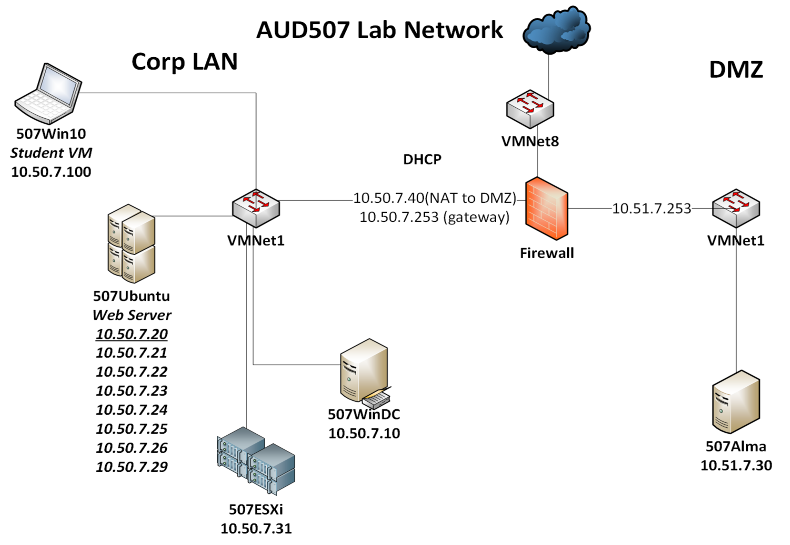
August 10, 2021

August 10, 2021

Technet24

© SANS Institute 2021

Exercise 3.4: Unix Logging, Monitoring, and Auditing

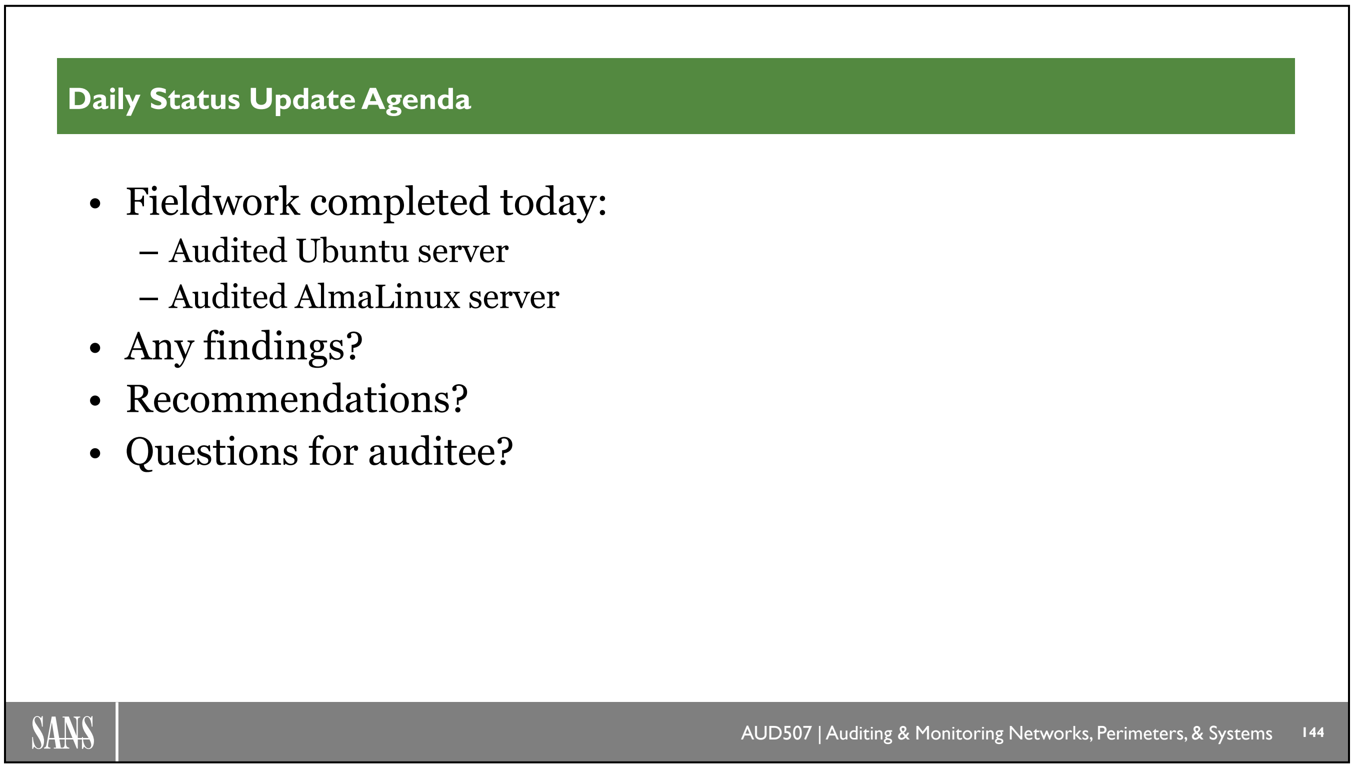


Please open your Workbook to Exercise 3.4: Unix Logging, Monitoring, and Auditing, or pick up with wherever you left off in the exercises.

August 10, 2021

August 10, 2021

© SANS Institute 2021



Daily Status Update Agenda

- Fieldwork completed today:
  - Audited Ubuntu server
  - Audited AlmaLinux server
- Any findings?
- Recommendations?
- Questions for auditee?

SANS AUD507 | Auditing & Monitoring Networks, Perimeters, & Systems 144

Think back on the fieldwork you completed today.

Were there any problems with patching on the systems? Are they nearing end of life? Was logging appropriate? Was Tripwire configured correctly?

August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24

© SANS Institute 2021

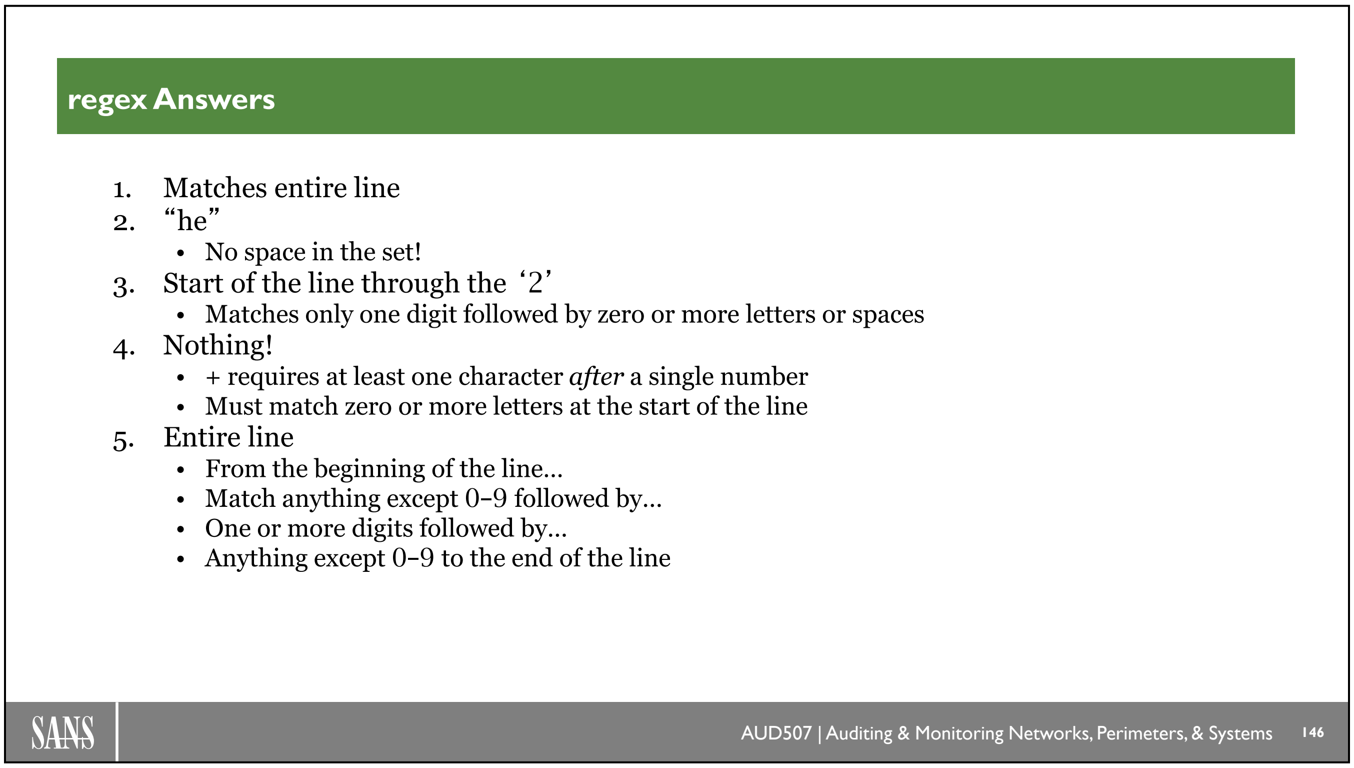
Thank You!

This brings us to the end of the Unix section. If you are taking the class at a conference, please take a moment to complete an evaluation form. You will be given a different evaluation for every section of the class.

August 10, 2021

August 10, 2021

© SANS Institute 2021



regex Answers

1. Matches entire line
2. “he”
   - No space in the set!
3. Start of the line through the ‘2’
   - Matches only one digit followed by zero or more letters or spaces
4. Nothing!
   - + requires at least one character *after* a single number
   - Must match zero or more letters at the start of the line
5. Entire line
   - From the beginning of the line…
   - Match anything except 0–9 followed by…
   - One or more digits followed by…
   - Anything except 0–9 to the end of the line

This page intentionally left blank.

August 10, 2021

© 2021 Risenhoover Consulting, Inc.

August 10, 2021

Technet24