



Introduction

Computer Network Defence - 2023/24

Version 1.01 22/08/2023

Purpose

This tutorial intends serves to setup and get the basic configuration in place for one of the key systems - a Linux Server. We will be using as a focus for securing and processing data. you will be installing a Linux server and configuring a number of services on it which we will use during the course. Some points to take note of:

1. We will be using Ubuntu Server 22.04.3 LTS as the base platform for the course. The server version is being used specifically as it is 'headless' - in other words has no GUI. This will help you develop a sense of realism as to how systems are actually operated an managed.
2. You will also need a client system to access the server. Something like Ubuntu/Mint desktop is recommended, but it is quite possible to do this directly with Windows/Ma-cOs (other than where directed).
The important thing is to have a system separate to your server bot to use as a 'cli-ent' for services to be run on the server, but also for using to perform administration on the server. Trying to work on the VM system 'console' is going to be slow, and painful.
3. You should install these in a suitable VMware environment as we will be adjusting the networking goring forward.

If you use a different platform, you are on your own! We will only support the systems detailed in this tutorial for the course

While often seen as the 'unsexy' side of security in comparison to Red Team work, Blue Team activities in many ways are more challenging, and can be extremely rewarding. A first step to getting your shields up and defensive posture in place is having a solid, and secure system on which to start from.

This course will be working extensively in the Linux environment. You need to be comfortable with moving around the system, `bash` (shell) scripting/automation and competent with use of basic commands. These are revised in Tasks 4, 5

During this course you will be required to develop a variety of configurations for software the Linux Server we will be building. You are **strongly encouraged** to keep a written notebook of your actions along with digital copies of the resulting files.

As a reminder, anything covered in the tutorial sessions, lectures and required readings is in scope for the online examination, and quizzes unless explicitly excluded.

Legend

We are trying a new tutorial template this year. To highlight the key areas in the document where you may need to download, read or answer questions we will make use of the following symbols/icons



You will need to do the following configuration



You may find this useful to answer the question, to get passed the current task, or just generally



You will need to download something from somewhere <http://somewhere.com>



You should answer this question



This will take a lot of time, but you will find it useful in addition to your study



This is additional reading that will support you and/or is needed to get further understanding of concepts.



This is audio and/or Video material for you to refer to.



Grab this project or resource from the following <http://gitub.com/awesomepotions>



This is a **POTENTIALLY RISKY task.**



You can do this to know you are correct

We hope you find these useful. This block may expanded or removed as we go on during the year.

This tut *may* be updated during the course of the week with additional support material. Make sure you have the latest version as shown on Moodle.


Contents

Legend	III
1 Preparation	1
2 System Setup	8
3 SSH Client Access	11
4 Linux Basics	12
5 BASH skills	14
6 Deliverable	17
Resources	17
A News	18
B Reading	19

CHANGELOG

- ➦ v1.0 - Initial Release
 - ➦ v1.01 - Fixed URLs in Preparation for 22.04.03
-

1 Preparation

For this course you will be installing, configuring up and securing a set of services on a Linux Server. The basis for this will be the  Ubuntu LTS¹ Server 22.04.3 Release. You need to start off by setting up the Server we will be using for the course.



Server Setup

You need to download the .iso for the latest current Ubuntu LTS server 22.04.3

- Download the ISO (ubuntu-22.04.3-live-server-amd64.iso) and ensure you can get this booted up and installed on your Vmware system. This can be downloaded from:
<https://releases.ubuntu.com/22.04.03/ubuntu-22.04.03-live-server-amd64.iso>
- Verify the SHA256 HASH
a4acfd10b18da50e2ec50ccaf860d7f20b389df8765611142305c0e911d16fd



Hash verification

- To verify that hash on Windows you can use the command:

```
CertUtil -hashfile ubuntu-22.04.3-live-server-amd64.iso SHA256
```

Make sure you are in the correct directory to run this or specify a full path.

- 🐧 To verify on a Linux system you can use the following command:

```
echo "a4acfd10b18da50e2ec50ccaf860d7f20b389df8765611142305c0e911d16fd
```

```
*ubuntu-22.04.3-live-server-amd64.iso" | shasum -a 256 --check
```

The response should be ubuntu-22.04.03-live-server-amd64.iso: OK



Meh, verification takes too long.. YOLO!

Why should one get into the habit of verifying downloads of operating system installers, packages, and software in general? What threats does this protect against? How could you enhance the integrity check?



Client Setup

You need to download a client system to use. This can be what ever you wish, but its strongly suggested to be a Unix like system. Examples include Ubuntu Desktop, Mint, or even Kali. You should however use something you are comfortable configuring and installing packages on.



While your software is downloading...

Make sure you have set your profiles up completely in Moodle, **and** on Teams

- Have you set a valid profile pic?
- Have you set your Name capitalisation correctly?
If you cannot change this, contact support

¹Long Term Service release - meaning its stable and gets updates for a extended period in this case 10 years from release until April 2032.



VMware Setup

We are using the VMware hypervisor to support the virtualisation needs in this course. This will allow us to install a virtual machine of our target system using VMware Workstation (or Fusion for those running Intel based Mac systems). It is strongly recommended that you set the system up to use **Host-only networking** (especially in a lab environment). While there is nothing at risk in this tutorial it's good practice to have minimum exposure of systems.

- Details for obtaining VMware Workstation/Fusion keys are available from the [Noroff Student Help Portal](#).
- You can log into Brightspace to download /retrieve your keys [here](#)
- VMware currently has support from Fusion 13 for running on newer Apple M1/M2 Mac products using Apple silicon. This is however still not recommended as there are some incompatibilities due to the way these CPUs support hypervisors. Support for running x86/x64 code became widely available only mid 2023. You are strongly recommended to use an Intel/AMD x84/x86 based system.



Should you wish to run another hypervisor platform you may, but we cannot provide any support on this.



Once you have downloaded the ISO, and have VMware running, you need to complete the following steps. These should take 20–25 minutes depending on the speed of your network and local system.

1. Install the Server in VMware. You should have at least 10GB of disk space for the virtual system. Follow the prompts and get the base system up and running (do not forget to create a user and set a password for it!)
2. Take note of the following configuration steps, and be sure to check your selections match those below:

```
Choose type of install [ Help ]

Choose the base for the installation.

(X) Ubuntu Server
    The default install contains a curated set of packages that provide a comfortable experience for operating your server.

( ) Ubuntu Server (minimized)
    This version has been customized to have a small runtime footprint in environments where humans are not expected to log in.

Configure Ubuntu archive mirror [ Help ]

If you use an alternative mirror for Ubuntu, enter its details here.

Mirror address: http://no.archive.ubuntu.com/ubuntu
    You may provide an archive mirror that will be used instead of the default.

SSH Setup [ Help ]

You can choose to install the OpenSSH server package to enable secure remote access to your server.

[X] Install OpenSSH server

Import SSH identity: [ No ]
    You can import your SSH keys from GitHub or Launchpad.

Import Username:

[X] Allow password authentication over SSH
```

- After the install is complete, restart, and ensure you can log in.
- Your server after its first reboot may look a little odd. Press Alt+F2 (switch to the second console) and log in. Its good practice **not** to log into the primary console, as this is where kernel error messages get dumped²

```

Ubuntu 22.04.1 LTS cnd22 tty1

cnd22 login: [ 19.775653] cloud-init[1551]: Cloud-init v. 22.2-0ubuntu1~22.04.3 running 'modules:final' at Thu, 25 Aug 2022 12:54:43 +0000. Up 19.71 seconds.
ci-info: no authorized SSH keys fingerprints found for user cnd.
<14>Aug 25 12:54:44 cloud-init: #####
<14>Aug 25 12:54:44 cloud-init: -----BEGIN SSH HOST KEY FINGERPRINTS-----
<14>Aug 25 12:54:44 cloud-init: 1024 SHA256:Lh5g89hi6+6qw04zxY10WnEjFd602/KCAd2fPSbHAg root@cnd22 (DSA)
<14>Aug 25 12:54:44 cloud-init: 256 SHA256:r3unj4K0IR/1eUL8uKra3JieM+uiNefaNwyfsUHTG6g root@cnd22 (ECDSA)
<14>Aug 25 12:54:44 cloud-init: 256 SHA256:3nafecmDWHGnzA4LREHwSPHCZ/NQicX4gbtri6E+8mI root@cnd22 (ED25519)
<14>Aug 25 12:54:44 cloud-init: 3072 SHA256:2yT8wzkWwXoMegLwZwetLsvzt1rW4jz1PvXdBtJFqg root@cnd22 (RSA)
<14>Aug 25 12:54:44 cloud-init: -----END SSH HOST KEY FINGERPRINTS-----
<14>Aug 25 12:54:44 cloud-init: #####
-----BEGIN SSH HOST KEY KEYS-----
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBLcMbU4TExnsFEC2mVxXgcxd725/WuSkokFvvQ/kPb8uS1T3Qr4j4o0kpwVf7gCKdHjX6QFM/qEgmp0oaz2FqA= root@cnd22
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIFdz14X+fG2AsAokYF1PFMG6HySaqwE5vLthVA92DspZ root@cnd22
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDQJ94nYzkJ3kPhpXhTFPP13Gv86oH5rtu5ja2lkfAtyJVniyRb3rNR6sKJOPdTPveQo1o8jfwNh4J4JP1Sgo13N0MBu6m8KMc0Uox1F1IzCH1q2IJzueb9G1x3eKt/M+J7R3EDPc42yxmENWzQxfRXAs+7iugXQTrK+f1LVMPeG42X25X/A2q/S2c06YBTauKSHozj9zoK5PYCOPcB1AwPnsJWM4G4ko1eshwNoqjCWDbyN4YsrntwSa97+w4X11GvQc1iNWKDXdGtGAdM8+2JmQXUgWB24H80iEb7nW94m0bG9q/1/IScCT2I4x+NxopwNtrUim98BJPLkTIXvTVL3SHTdGQ3UNqmmjx4EunpgrMsxU9yJ3Z1390M7UKim0jogCLTY8lefqqpMnT/ogHQ2q001bD+hAcHHgfHAvhQmLzPAHS1edi8uiv08vyERTvA5U6Dt0/DdFVQHgJIXE53VhKEVIUY+5FLD01g64rBtTQz+gT5tr2GpX9vn9fP1TU= root@cnd22
-----END SSH HOST KEY KEYS-----
[ 19.867700] cloud-init[1551]: Cloud-init v. 22.2-0ubuntu1~22.04.3 finished at Thu, 25 Aug 2022 12:54:44 +0000. DataSource DataSourceNone. Up 19.86 seconds
[ 19.868265] cloud-init[1551]: 2022-08-25 12:54:44,089 - cc_final_message.py[WARNING]: Used fallback ck datasource

```

- After you successfully log in, you should see something similar to the screen below.

```
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-78-generic x86_64)
```

```

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

```

```
System information as of Tue Aug 15 01:48:45 PM UTC 2023
```

```

System load:  0.04638671875    Processes:            214
Usage of /:   26.8% of 9.75GB   Users logged in:      1
Memory usage: 9%               IPv4 address for ens33: 192.168.15.129
Swap usage:   0%

```

```

* Super-optimized for small spaces - read how we shrank the memory
  footprint of MicroK8s to make it the smallest full K8s around.

```

```
https://ubuntu.com/blog/microk8s-memory-optimisation
```

```
11 updates can be applied immediately.
```

```
To see these additional updates run: apt list --upgradable
```

- Use the `sudo` command to elevate your user's privilege level to that of 'root', and then check for any system/software updates that need to be applied. This can be done using the `apt` command.

²By default, Ubuntu Server provisions six consoles `tty1` though `tty6`. You can access these using Alt+F1 though Alt+F6.

7. Apply any updates, reboot and check you can log in. Shut the system down at this point and create a snapshot of the VM. This will allow you to go directly back to this point in the future if you make your system unusable.

Accessing Your Server

During the course of the tasks you will be undertaking in the coming weeks, there will be a significant need to be able to copy and paste material to and from your system. In addition some commands will produce output that goes beyond a single screen. Doing this at the console within VMware can be difficult. A far better way is to make use of ssh to remote login to the server (you did check it was selected at install time?)³.

Using ssh from a 'remote' system either from the command line or via a dedicated client allows you to have multiple connections in parallel to your server, too allow for monitoring as well as copy paste between them, and the notes you should be keeping and/or the task sheets. ssh is discussed in Tasks 2 and 3.

To access your server remotely over the network (even though it is a virtual network on your machine) requires that you know the IP address to connect to. Fortunately the default *motd*⁴ includes the IP address when you login in as:

IPv4 address for ens33: 192.168.15.129



Connection Test

Before you move on make sure the following are done:

1. Check you can ping the IP address of your new server VM from your host machine.
2. Check you can ping your host from the VM. On Windows and Linux desktops default configurations should allow this, but you may need to check your host firewall if this does not work.

If you are having issues, *now* is the time to debug your network.

On systems where this is not the case you will need to use the `ip` or `ifconfig` commands to find it out.



Bad Habits: working on console

You **will** find it very difficult working directly on the console. The console should be used for initial configuration and for logging into the system directly should you lock yourself out or remote access crash. This is in addition good practice to get used to where in industry, servers are typically in a datacenter in a different building, city or even country!



Working with your VM

You may find the VM system easier to use if you install the `net-tools` package which provides command such as `netstat` and `ifconfig`. You **will** need to upload files to your Ubuntu VM. Filezilla⁵ for Windows system you can use `putty`⁶, `Bitvise`⁷ or similar client to connect from your host system to the VM via the console. Both Filezilla and `putty` are available on Ubuntu/Debian/mint systems and can be installed via `apt`. This can be useful when working with lots of data, or wanting to use multiple screens.

When working on the text based Task, it may be much easier to have a ssh session connecting to your host (especially with large or wide screens) as you can easily cut and paste from this document, web pages and your own notes – something that can be tricky sometimes with the Virtual machine.

³If you forgot to select it, you can install the `openssh-server` package with `apt` and reboot.

⁴`motd` - Message of the Day, is what by tradition Unix systems display when a user logs in. It is located in `/etc/motd` and more can be found out using the command `man 5 motd`

⁵<https://filezilla-project.org/>

⁶<https://www.putty.org/>

⁷<https://www.bitvise.com/ssh-client>



Network Tools

You may find the server easier to use if you install the `net-tools` package which provides command such as `netstat` and `ifconfig` alternately you can use the build in `ip` command⁸. The following will return interfaces with their corresponding ipv4 addresses:

```
ip -4 addr
```

Making your life easier

While as shown above it is possible to get an IP via logging in, you can make a few modifications to the server to make the process much easier. By default the message on your VM console should look as shown below:

```
Ubuntu 22.04.1 LTS cnd22 tty4
cnd22 login:
```

While perfectly serviceable, it does not contain a lot of information that is of use to us. Fortunately this can be remedied by editing `/etc/issue` which is the file displayed by the `getty` terminal manager (i.e. the application that enables you to log in).



\$EDITOR is best!

The debate as to which is the best editor to use on unix system is likely to raise fierce debate⁹. By default, the ubuntu server ships with two editors installed `vi`¹⁰ and `nano`. Many new users may find `nano` easier to use. It is important that you are able to use an editor *on* the server, rather than editing and uploading files from your host. Where are a wide range of editors available and you should use editor of your choice. Additional editors can be installed with `apt`. If you are going to use `vi` a more fully featured version (with colour highlighting) is `vim` and can be installed using the `vim-tiny` package. You will have to remember to type `vim` however.

Some tools such as for changing user details rely on the system default editor. This is defined on a per user basis using the `EDITOR` environment variable. If you try a tool and get such in something that looks like `vi`, work out where your chosen editor is. For example to use `nano` type `which nano`. This will tell you where the editor lives on the file-system. The `EDITOR` environment variable can be set using the command `export EDITOR=/usr/bin/nano`. This is then set in your environment.

If you edit `/etc/issue` and add a second line "IP: \4" you should see an improved version that now displays the system IP address. This means you can see the IP at boot without having to log into the server. The \4

```
Ubuntu 22.04.1 LTS cnd22 tty3
IP: 192.168.15.129
cnd22 login: _
```

⁸This command is slowly replacing the more traditional `ifconfig` that has been standard across unix-like systems - Linux, SunOS, Solaris and the *BSD family. Its worth getting to know the basics of it

⁹Why this happens is unclear, `vim` is obviously the best! Some people do seem to like the absurd complexity of `EMACS` although in the early days of unix a different acronym was more apt

¹⁰Many people are terrified of `vi` and its successor `vim` but its worth getting to know as it is a core editor available on most Unix-like platforms, where the likes of `joe`, `nano` and `ee` may not be. The most important command is how to **quit**: `ESC :q!`

More details on using it can be found [online](#)

is interpreted at runtime and expanded out as the IPv4 address of the system.

If this is *not* displaying when you switch to an unused terminal (eg Alt+F3), you can go to another terminal and press CTRL+D (which simulated a logout event) and will cause the software to reload. alternately logout, or reboot (but these should not be needed). There is quite a bit of functionality that you can extend. more can be found in the manual page for `getty`.

Using the details from the manual page, edit your `/etc/issue` so that it produces output such as that below.

```
Computer Network Defence 2022/23
Ubuntu LTS cnd22 tty3
Terminal start: Thu Aug 25 2022
IP: 192.168.15.129
cnd22 login:
```



The `man` command is of great use when you want to know how to do things on a unix platform. This can be used to get the manual page for a specific command – for example `man man`.



Another way of determining command line parameters for a command is running it with the `-h` or `--help` flags, and seeing the summary that most commands return.

Before You Continue



Ready to Go

Before you move on make sure the following are done:

1. You have the VM installed
2. You can log in, and use `sudo`
3. You have applied updates, and rebooted
4. You had tweaked `/etc/issue` to make your life easier
5. You have checked the new issue works, and shows up on boot.
6. You have taken a snapshot of the VM to roll back to should (when?) you break anything.
7. You have uploaded a valid picture of you on Teams
8. Is your name set correctly on Moodle, especially capitalisation?¹¹
9. Have you set a picture of **you** on Moodle¹².

When you have your base system up, you should continue with the tutorial tasks that follow.

¹¹Unless you are [ee cummings](#), names are normally capitalised.

¹²While purple bunnies are great, its better to think of you as a person rather than a animated entity ;-). It further helps staff get to know you.

**Snapshots**

Snapshots do not make up for a proper backup strategy, but they do offer flexibility and the ability to roll back your Virtual Machine to a prior state. You are encouraged to use this feature before making substantive software or configuration changes; or where there is a risky operation. Snapshots do however use a little bit of disk space - how much depends on how much has changed on your system. You can read more about the snapshot process at [VmWare](#)

2 System Setup

Now that your system has been setup, you need to complete a few additional points around the installation

Adding a Backup User

We will be working with different authentication methods during the course. As such its always useful to have a second user you can use as a backup, if you run into authentication problems, and/or lock your primary user out. These are both things that are quite possible as we start editing configurations. You must undertake the following steps:

1. Create a new user called `recovery`. You can use the `adduser` command to do this. The output should look similar to the text below.

```
Adding user 'recovery' ...
Adding new group 'recovery' (1001) ...
Adding new user 'recovery' (1001) with group 'recovery' ...
Creating home directory '/home/recovery' ...
Copying files from '/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for recovery
Enter the new value, or press ENTER for the default
    Full Name []: Recovery user
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] y
```

2. Check that the user has been created, and what permissions they have in comparison to your initial user. In the example below, the initial user created is `cnd`.

```
cnd@cndtut:~$ id cnd
uid=1000(cnd) gid=1000(cnd) groups=1000(cnd),4(adm),24(cdrom),27(sudo)
cnd@cndtut:~$ id recovery
uid=1001(recovery) gid=1001(recovery) groups=1001(recovery)
```

3. As you can see there is a difference. Try open up another console (Alt+F3) or ssh session and log in as the `recovery` user. What happens when you try use `sudo` ?
4. Have a read as to how `sudo` and the `/etc/sudoers` file work using the `man` command. Using this information, make the appropriate changes so that the `recovery` user can use the `sudo` command, and gain root equivalency.
5. Make sure you remember the password for this user¹³.

¹³Getting back into a system where the password has been forgotten is not particularly difficult in most cases, and may not even require additional media, only physical console access. This is another reason why PHYSICAL SECURITY of systems is so important.

Server SSH Keys

Much of the access to the server system will be via SSH¹⁴. This offers a number of distinct advantages over trying to work on the console:

- You can have multiple connections. Think of each window as a separate connection to the system.
- You have 'remote' access to the system.
- You can easily pull up different sessions side by side. This is useful when you need to watch log as you are configuring a system
- Duplicate sessions can help prevent you locking yourself out.
- Your sessions are encrypted, and authenticated.



What other benefits can you think of in using SSH to connect to the server ?

In order to be able to create a ssh connection to the server you will need an appropriate client. This typically comes pre-installed on unix platforms.



If you are using SSH on Windows two popular options are:

- PuTTY <https://www.putty.org/>
- BitVise SSH client <https://www.bitvise.com/download-area>

Both of these allow for easy cut and paste of content to and from the server to your notes. Be careful when pasting in commands from elsewhere. Typing things in is generally slower but less changes for things to go wrong while you are learning.

Microsoft also provides some [guides](#) if you wish to install OpenSSH (as used on Unix systems) natively on Windows.

The first step to being able to trust your connection is to verify the cryptographic keys that were generated as part of your server install. These files (referred to as server SSH keys) are stored in `/etc/ssh`. Each of these files is used for a different cryptographic algorithm. We will look later on at how to configure these. An easy way of generating the fingerprints for these files can be done as follows:

```
~$ for file in /etc/ssh/*key.pub
> do
> ssh-keygen -lf $file
> done
1024 SHA256:bIZ6Py89djSA1MNvTSeAId6MMFaf9qL/LJ8hfdbFE0Q root@cndtut (DSA)
256 SHA256:P00Wm7bMtH7LbBPmtG7HrEXTJT+4WKkgP6a0ievltkU root@cndtut (ECDSA)
256 SHA256:JKn2aL3JkRYX2qw6JeoKIMFHhy7Dzmtsb7dklySGBfU root@cndtut (ED25519)
3072 SHA256:qAXZFd4A8EUbZQ9kXmt60j9ri0hGATXPeRgaFityw2c root@cndtut (RSA)
```

Knowing what the key *hash* or *fingerprint* is is an important for being able to trust a first connection to a server from a client. Connect to your server, and verify the key fingerprint presented. On Unix systems this is then stored in the file `~/.ssh/known_hosts` (ie. in the `.ssh` subdirectory in your home directory). Any future connections to this host are verified against the saved fingerprint.



If these files are **not** present you need to ensure that the `openssh-server` package is installed. and then restart the machine.

¹⁴[https://en.wikipedia.org/wiki/Ssh_\(Secure_Shell\)](https://en.wikipedia.org/wiki/Ssh_(Secure_Shell))



Getting a server fingerprint

You can retrieve fingerprints for a server by running the command:

```
ssh-keyscan hostname
```

This can be compared to lines in the `known_hosts`. The easiest way to do this is to use the command `ssh-keygen -F hostname` to recover the stored key for a given host

SSH Keys in DNS

One way of helping distribute and validate SSH key *fingerprints* is by putting them into the Domain Name System DNS. An example can be seen for the host `trollhelm.moria.org`. These commands need to be run on a Linux system (such as your server)

When issuing the command below, the various SSH key fingerprints are shown:

```
$ host -t sshfp trollhelm.moria.org
trollhelm.moria.org has SSHFP record 2 1 9045813476B58919C912AAD35177AB0A0A7C03E2
trollhelm.moria.org has SSHFP record 3 1 088D8B2C0F8C6D5CCD7FC9CFF6246EA583BB7116
trollhelm.moria.org has SSHFP record 1 2
    EA87962FA9A79C8F707CB05319A2B31E63F81BA8F70E37AAE7FFCCE6 B96A6EBD
trollhelm.moria.org has SSHFP record 3 2
    E18FFC6C0272E5EF4171F3B23816BC686D69BD52405E4FA01BFEAECD 2DD7C4F4
trollhelm.moria.org has SSHFP record 4 1 58ED8199902B7DA08419C3C79ABC03D1E2890C0F
trollhelm.moria.org has SSHFP record 2 2
    72971404576EE0A3FA7CA2FF595346C94CE4C42A72718986FF089D6C 021FB7D1
trollhelm.moria.org has SSHFP record 1 1 1A091E48F073E31DD5BC8B738FCA6BC3828304F6
trollhelm.moria.org has SSHFP record 4 2
    F0641C4BA0349EB065343402FF8055D59BBC61A8CECC1ED3C4DFC80A 6B3D67D3
```

You can instruct your own SSH client to verify any keys it receives with what is available via DNS. An example run from a Linux shell, using the same host as before would be:

```
$ ssh -o VerifyHostKeyDNS=ask bvi@trollhelm.moria.org
The authenticity of host 'trollhelm.moria.org (95.216.184.98)' can't be established.
ECDSA key fingerprint is SHA256:4Y/8bAJy5e9Bcf0yOBa8aG1pvVJAXk+gG/6uzS3XxPQ.
Matching host key fingerprint found in DNS.
Are you sure you want to continue connecting (yes/no/[fingerprint])? fingerprint
```



Can you determine what the different values for the SSHFP DNS records are?



- How would you configure your ssh client to automatically verify server keys in DNS?
- What do you think a downside of this could be?
- How could you mitigate it?
- What commands could you use at the Windows command line for retrieving this information?

3 SSH Client Access

SSH keys are not just for servers. They can be used to securely authenticate your own connection to a server. You need to create a set of SSH keys of for **both** your user and the *recovery* user, and ensure you can use the appropriate private keys to log into the server from your client system(s).

While generally not good practice to have private keys on the server (unless there is a specific need), for the purposes of this tutorial its is suitable to generate they ssh keys on the server. You will need to use the `ssh-keygen` utility. See its manual pages for more details.

Alternately you can generate the keys on the client, and copy up only the public portions. For production use you should **always** protect these keys with strong passwords, as anyone who has a key for a user can log into a server as that user.

This exercise also provides an opportunity for you to practice getting files to/from your server and host. While keys can be set up using copy and paste, you will need to move files around for later tasks.



SFTP and SCP are two utilities that make the copying of files much easier between systems. GUI options are included as part of the BitVise SSH client. A recommended alternate is to use [Filezilla](#) for moving files such as the private key id around.

If you get stuck here are some additional guides (they still apply to the newer Ubuntu version):

- [How to Set Up SSH Keys on Ubuntu 20.04](#)
- [How to Set Up SSH Keys on Ubuntu 20.04](#)
- [How to Set Up SSH Keys on Ubuntu 20.04](#)



Ready to Go

Before you move on make sure that you can do the following:

1. Have a password/passphrase to protect your private keys.
2. Have verified that they keys work both for your primary and *recovery* user.
3. Can log in using the private key to the server. Use `ssh -v` to verify this from the command line. If you are using a gui client turn on appropriate debugging.
4. Can inspect the SSHd (server) logs on the VM server system and see what the difference is between password and key based logins looks like.

4 Linux Basics

NOTE: this is not a required task, but is here to remind people who have got a little rusty with Linux over the summer and/or since the Operating Systems course.

The most prevalent operating system you will use and be exposed to as a security analyst is Linux. Although operating systems such as Windows and MacOS have some powerful tools that can be used — and are used very successfully - Windows in particular is not fully equipped with tools needed in its default state. It is important that you are comfortable working in Linux. Unix skills are important particularly in the field of web application penetration testing, as the vast majority of web hosting runs on these platforms.

This course will be working extensively in the Linux environment. You need to be comfortable with moving around the system, and use of basic commands such as `awk`, `sed`, `grep` and `bash` (shell) scripting. These are covered in more detail in tasks following.

This serves as a quick start to navigation and working at the command line. Even if you are familiar with Linux or Unix systems, it is worth ensuring you have covered all the aspects covered below. Throughout this week we will work on Linux OS Basics. Use the Virtual Machine (VM) you installed in the Preparation section. For this task, work through the tutorial below. The order and rate at which you progress depends on your current familiarity.



If you feel comfortable and can do the required items in the list below, proceed with the next task, but you can always come back here to refresh your skills.



Levelled up?

A simple set of metrics for you to check against before proceeding to the main tasks for the course are to evaluate if you are able as a *minimum* to do the following:

1. Navigate through the file system at the command line
2. Create, move and delete directories
3. Produce a simple script that can run a series of commands
4. Use a *loop* to run commands against a series of files, or lines of a file
5. Pass input from a file, redirect output to a file, and pass data between commands using standard input and output operators of `<`, `>` and `|`
6. Be able to create and edit text files.
7. Connect to your system remotely - both with a shell (`ssh`) and for file upload (`sftp`).
8. Be able to copy files up and download files to your system.

A very good tutorial for learning the basics Linux OS is available at:

<http://www.ee.surrey.ac.uk/Teaching/Unix/>

This tutorial can be used in combination with a wide variety of online resources to support your tasks. One of the reasons that Ubuntu has been chosen as the base OS is its widespread support resources available online.

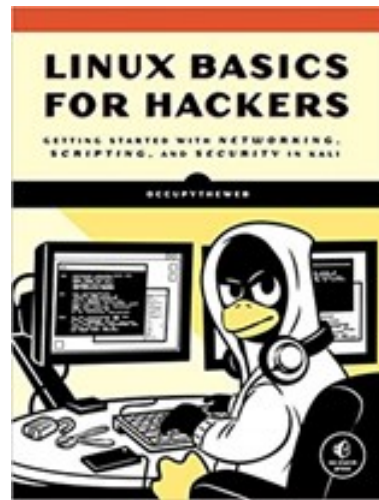
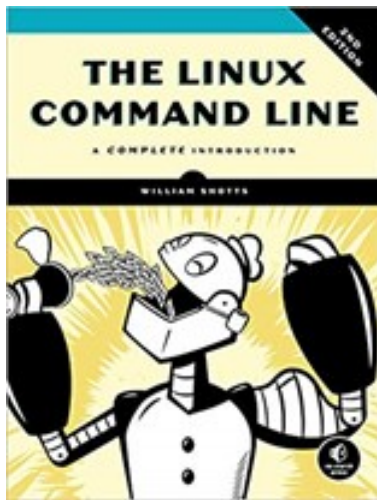


You can work on an additional tutorial around access rights and file permissions here:

<https://ryanstutorials.net/linuxtutorial/permissions.php>



If you want to continue building on your Linux command line proficiency, the following resources are recommended:



[Humblebundle.com](https://www.humblebundle.com) often has very good collections of ebooks available are worth keeping an eye on as you build up your own professional library.

5 BASH skills

A large part of doing security analysis is being able to automate your activities, and process data. Sometimes Python is overkill for the task at hand (or unavailable on a target system). As an alternative the Bourne Again Shell — (BASH). While there exist many different shells that can be used (including a port of Microsoft's powershell¹⁵) they share a common heritage. Minor syntax differences do exist which is why its important to specify at the start of a script which shell its to be run under (or do so on the command line!).

A very good tutorial is available at:

<https://linuxconfig.org/bash-scripting-tutorial-for-beginners>

For this task you should use the VM you installed in the Preparation section. This activity is **highly recommended** even if you have some exposure to Unix/Linux as it covers a number of aspects not commonly used.



While the temptation is to do everything as a privileged (root) user, this is courting disaster. This is partly why the Debian family of systems make use of the `sudo` command. Especially when learning accidents can happen and you can damage your system. Run commands as a non-privileged user.

Hint: the command prompt by default tells you if you are in privileged mode, changing from a '\$' to a '#' as shown below.

```
cnd@cndserver:/data/$ sudo -s
[sudo] password for cnd:
root@cndserver:/data/#
```

The suggested approach to working though this this week is as follows.

1. Work though this tutorial up to Variables
2. Complete up to Numeric and String Comparisons. Once you grasp the concept of these move on. You can always come back here as a reference when you need to find out or do specific comparisons.
3. Conditional Statements and Loops should be fairly familiar from Python. These are identical concepts, the syntax just differs slightly.



Regarding the looping constructs, **for** and **while** are the most commonly used, and useful tools to use. Make sure you can use these **both** at the command line and in a script.

4. Positional argument are one of the areas people often struggle with, but are really one of the most power features of scripting. Using these in your scripts allows for great flexibility as you can control what the script does (and to what) based on arguments (parameters) you pass on the command line.
5. Although arithmetic is not used that often it can be very useful on occasion. By this stage you should be comfortable integrating other Unix commands into scripts to automate a series of steps. you can always refer back here for details when you need it.

¹⁵See [Microsoft Guide](#) and [Linux Guide](#) for details on installing this.

**What to know?**

It may be a challenge trying to remember all the new elements you have learned. Fortunately you are not expected to be an expert (or even comfortable) after one week. We will be building on these skills though the course and pointing out tips and tricks. Two key areas that will help you are:

1. Keep notes. Keep copies of your scripts, both on the system, and in a separate text/-word/L^AT_EX document. Build yourself a cheatsheet.
2. Know where to find information. In many cases it is more important to know that information exists and you can reference, than being able to memorise everything. When you know where information is (or can use Google effectively) you can refer back to it as needed. Eventually you remember what you use commonly and look up the rest as you need it.

There is however no substitute for the process of *drill and practice* - active doing and repeating tasks.

You should use your judgement around this ordering depending on your skill level. If you get stuck in later tasks, or weeks, come back here.

**Levelled up?**

A simple set of metrics for you to check against to evaluate if you are able as a *minimum* to do the following:

1. Read in a text file and do something (echo) every line
2. Be able to do something to every file in in a directory
HINT: of all the loops the *for* loop is by far the most flexible and useful for iterating over a list of items. We will use this frequently in the course.
3. Produce a simple script that can run a series of commands.
4. Use a *loop* to run commands against a series of files, or lines of a file
5. Be able to input from a file, redirect output to a file, and pass data between commands using standard input and output operators of `<` , `>` and `|`

See Task 6 for a final task bringing your skills together.



1. What do you feel the value is of being able to automate tasks?
2. What is an advantage of being able to have a process contained in a script rather than a list of commands an analyst must run?
3. What is one drawback you can think of regarding automation?
4. Why is this important to a security analyst?



Some additional resources you may find valuable are:

- [BASH Programming - Introduction HOW-TO](#)
- [Bash scripting cheatsheet](#)
- [Bash Guide for Beginners](#)

Video Resources

Some of you may prefer video tutorials. The following are a small selection, and you may well be able to watch them sped up. When using these make sure you cover the same topics as in the tutorial text at the start of this Task.

- [BASH Scripting EP 1](#) ⌚ 12:13
- [BASH Scripting EP 2 \(conditions\)](#) ⌚ 29:42
- [BASH Scripting EP 3 \(loops\)](#) ⌚ 21:07
- [Beginner's Guide to the Bash Terminal](#) ⌚ 1:14:36
- [Write Your Own Bash Scripts for Automation](#) ⌚ 15:34



Unix (and thus Linux) has a very strong philosophy of **one** task, **one** program. This allows one to string commands together to achieve quite complex outcomes made up of simple parts. We will explore this next week with some Text Processing.

- Think about the benefits of this approach rather than large monolithic applications that do a lot of things.
- How does the system of files and file manipulation/redirection help with this?

6 Deliverable

At the end of this practical sheet you should be able to use your skills to solve the problem below. You will be required to have these available for the *Engagement task* in Week 3.

You should generate the following scripts:

1. The script should take a username as a parameter. When run it must ask you for your name and print this out along with the supplied username. Example output is shown below:

```
Your name: John Smith
Hello John Smith, your username is cnddemo
```

2. The script must take a filename as a command line parameter. It then needs to calculate the sha256, md5 and sha1 checksums over the file, placing one per line. The results should be written to `filename.CHECKSUM`. This should then be printed to screen, followed by its size in bytes before exiting. Example output below:

```
116578dda0e03f1421c214acdd66043b586e7afc7474e0796c150ac164a90a2a  test.iso
98213481fef82b9337de96469edb5089  test.iso
6e50e3288c5db5deaf2c977e3a7f541c80a4f971  test.iso
244 test.iso.CHECKSUM
```

3. The script should process all text (*.txt) files in a directory. For each of these files it should calculate either the size in KB **or** the md5sum. Which of these behaviours it undertakes is the same for all the files, and is defined by the first command line argument which can be either: **size** **or** **sum**. As processed each file it should provide a status report by printing out the filename. The script should inform the user what processing it is using.

Bonus: Can you extend the above to Print the filename before processing it, and then OK on the same line when processing is complete.

In addition to the above ensure that the following are complete:



1. Your Server VM is installed and updated
2. You have created users as required
3. You have a client system set up
4. You can transfer files to/from your client and server
5. You have SSH login working along with copy and paste.
6. You have worked through the linux and Bash tutorials
7. You have made backups!

A News

The following are new articles of interest in the last week. These are all relatively short, and you should get in the habit of staying aware of what is going on in the industry at large. The topics discussed here fall in scope for the engagement tasks.



News: Week of 21 August 2023

- MALWARE** [Add 'writing malware' to the list of things generative AI is not very good at doing](#)
"Despite the hype around criminals using ChatGPT and various other large language models to ease the chore of writing malware, it seems this generative AI technology isn't terribly good at helping with that kind of work."
- APT** [APT29 is targeting Ministries of Foreign Affairs of NATO-aligned countries](#)
"Researchers uncovered an ongoing spear-phishing campaign conducted by Russia-linked threat actors targeting Ministries of Foreign Affairs of NATO-aligned countries."
- ATTACK** [Beware of New Hacking Attack Targeting LinkedIn Accounts Worldwide](#)
"An ongoing campaign has resulted in the compromise of multiple LinkedIn accounts. Numerous users have reported instances of their LinkedIn accounts being temporarily locked, hacked, or permanently deleted."
- PRIVACY** [Why you should delete old accounts you no longer use](#)
"In all your time online, you've most likely signed up for hundreds of online services or platforms you no longer use or have forgotten about completely. Now, while you may not recollect these endeavours, the accounts may still be around, leaving any personal information they may contain up for grabs to any opportunistic or malicious party."
- VULNS** [Experts Uncover Weaknesses in PowerShell Gallery Enabling Supply Chain Attacks](#)
"Active flaws in the PowerShell Gallery could be weaponized by threat actors to pull off supply chain attacks against the registry's users."



What are you reading?

- Where do you get your Infosec/Cybersec news?
- How do you keep up to date with happenings in the domain?
- Do you regularly read and keep up to date with industry trends?

Post any good resources on Teams.

B Reading

Required Reading for the week:



- Ross Anderson (2020) [Who is the Opponent?](#) which is Chapter 2 in *Security Engineering 3rd ed.* Wiley; ISBN 1119642787.
This is [available](#) via the authors site. Estimated read time is 45-60 minutes.
This **does not** need to be all read this week, but should be complete by week 3.

Access Controls

Computer Network Defence - 2023/24

Version 1.1 29/08/2023

This week, we focus on ACCESS CONTROL (of which AUTHENTICATION is an important part) and the technical and theoretical aspects that relate to it. The lecture provides an overview of the challenges and approaches to authentication. These theoretical concepts are revisited in the primary reading for the week.

This course will be working extensively in the Linux environment. Last week you refreshed your skills relating to moving around the system, doing some basic shell scripting and use of basic commands. You expand your skills this week focusing on using built in tools such as `awk`, `sed`, `grep` for manipulating text in Task [5](#) and the Deliverable.

With the base systems installed, next week will continue to build on applying skills to security data and systems.

The practical tasks this week are made up of four distinct groupings:

SETUP *[Tasks 1-3]* You continue to provision basic services for our server. You will be adding a LAMP stack and Web application. These services need to be in place in preparation for next weeks tasks where we will be extending, exploring and securing these services. Task 1 in particular should serve largely as a refresher on public key cryptography, and the PGP ecosystem. Use your own judgement as to how comfortable you are with the concepts and skip to the end of Task 1 if you are.

CONFIGURATION *[Task 4]* Considering authentication and particularly strong authentication, you configure your server to support Two-factor authentication, in combination with a mobile authenticator app, to protect system logins. The second part of this activity explores some of the configuration options within SSH, with a primary focus on Access Control, and being able to be selective in how you apply it to incoming authentication requests.

APPLICATION Continuing to build your skills, there Task 5 introduces you to processing text files at the command line. This is a fast, portable and scalable way of working with large volumes of data such as log files. The Deliverable for the week provides a further opportunity to exercise your skills.

READING There are five brief news articles to review, along with a reading.

These areas can all be worked on independently, with no inter-dependency between the sections. Only the final **APPLICATION** section requires exploratory problem solving. The first two areas (four Tasks) of are comprise largely following instructions, and validating configurations. The order in which you approach these four areas is up to you. You are strongly encouraged to switch between tasks if you get stuck or find your concentration drifting. Task 5 provides a walk-through of the activity and processing needed to solve the challenge in the Deliverable.

Legend

We are trying a new tutorial template this year. To highlight the key areas in the document where you may need to download, read or answer questions we will make use of the following symbols/icons



You will need to do the following configuration



You may find this useful to answer the question, to get passed the current task, or just generally



You will need to download something from somewhere <http://somewhere.com>



You should answer this question



This will take a lot of time, but you will find it useful in addition to your study



This is additional reading that will support you and/or is needed to get further understanding of concepts.



This is audio and/or Video material for you to refer to.



Grab this project or resource from the following <http://github.com/awesomepotions>



This is a **POTENTIALLY RISKY task.**



You can do this to know you are correct

We hope you find these useful. This block may expanded or removed as we go on during the year.

This tut *may* be updated during the course of the week with additional support material, make sure you have the latest version as shown on Moodle.

Contents

Legend	III
1 PGP/GPG	1
2 LAMP Stack Setup	7
3 WordPress	10
4 MFA	12
5 Text Processing	15
6 Deliverable	24
Resources	25
A News	26
B Reading & Listening	27

CHANGELOG

1. v1.0 - Initial Release
 2. v1.1 - Warning added to Task1; added tips in Task 2; url fixed in Task 3; Warning added in Task 4.
-

1 PGP/GPG

The GNU Privacy Guard (GPG) is an open re-implementation of the Pretty Good Privacy (PGP) application originally developed by Phil Zimmermann developed PGP in 1991, during the height of the [cryptowars](#) in the USA and in Western Europe. PGP/GPG is used for signing, encrypting, and decrypting texts, e-mails, and files, to increase the security of data and communications. GPG follows the OpenPGP standards¹ as defined by the IETF in [RFC4880](#).

You have covered these principles previously in earlier courses so this should serve largely as a refresher to the technology, and as preparation for us needing strong encryption for this course.

Setup

There are two setup components to follow one for your server, and a second for an installation on a Microsoft Windows client.



GPG is generally installed as part of the base Ubuntu operating system (as it is used to verify the integrity of the repository indices and individual packages) as `gpg`. You can verify this by running `gpg --version` which should give output similar to that shown below. If this is not installed you will need to use `apt` install the `gpg` package.

```
gpg (GnuPG) 2.2.27
libgcrypt 1.9.4
Copyright (C) 2021 Free Software Foundation, Inc.
License GNU GPL-3.0-or-later <https://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: /home/cnd/.gnupg
Supported algorithms:
Pubkey: RSA, ELG, DSA, ECDH, ECDSA, EDDSA
Cipher: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
        CAMELLIA128, CAMELLIA192, CAMELLIA256
Hash: SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression: Uncompressed, ZIP, ZLIB, BZIP2
```



The [GPG4Win](#) distribution (currently version 4.0.3) is one of the more popular and free solutions for Windows platforms. It provides a number of integration plugins for email clients and browsers. The primary interface to this is the KLEOPATERA client from where you can generate and manage keys.



Make sure you understand the difference between **signing** a file vs. **encrypting** it.



Read this article on [Understanding Encryption, Signing and Verification](#) if you need a refresher on/or more clarification of the processes.

¹[OpenPGP](#) has a number of related standards - notably adding additional algorithm support

Generation



The following applies to the use of `gpg`

While its convenient to generate and manipulate keys at the command line it is not the only way. If you use an alternate tool you will need to follow the appropriate steps for that. Use of the command line is **not** mandatory, but you need to ensure that you can undertake **all** the operations required using your chosen tool as these will be needed later in the course.

You need to generate a PGP key with the following properties:

1. Use your real name and your @stud.noroff.no email address
2. That can be used for Signing **and** Encrypting

You need to start by generating a new key for your user. A quick-start can be done using `gpg --gen-key` to access all the options you can use `gpg --full-generate-key`. The result of going through this process should look similar to the output below. note the screen will clear when prompting you for your password.

GnuPG needs to construct a user ID to identify your key.

Real name: CND Demo user 2022

Email address: cnd2022@noroff.no

Comment: DEMO KEY for CND course 2022

You selected this USER-ID:

"CND Demo user 2022 (DEMO KEY for CND course 2022) <cnd2022@noroff.no>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? 0

We need to generate a lot of random bytes. It is a good idea to perform some other action (type on the keyboard, move the mouse, utilize the disks) during the prime generation; this gives the random number generator a better chance to gain enough entropy.

We need to generate a lot of random bytes. It is a good idea to perform some other action (type on the keyboard, move the mouse, utilize the disks) during the prime generation; this gives the random number generator a better chance to gain enough entropy.

gpg: /home/cnd/.gnupg/trustdb.gpg: trustdb created

gpg: key 0EB19529D7C1F21D marked as ultimately trusted

gpg: directory '/home/cnd/.gnupg/openpgp-revocs.d' created

gpg: revocation certificate stored as '/home/cnd/.gnupg/openpgp-revocs.d/EFD62D9A07C4751B42B3BD9C0EB19529D7C1F21D' public and secret key created and signed.

```
pub  rsa3072 2022-09-01 [SC] [expires: 2023-09-01]
     EFD62D9A07C4751B42B3BD9C0EB19529D7C1F21D
```

```
uid          CND Demo user 2022 (DEMO KEY for CND course 2022) <cnd2022@noroff.no>
```

```
sub  rsa3072 2022-09-01 [E] [expires: 2023-09-01]
```

This should leave you with a number of files in your `~/.gnupg`²



WARNING: Anyone who has access to your .rev file for a key can revoke your key, and thereby declaring it invalid. Protect it appropriately!

²The tilde (~) symbol is used as shorthand in Unix systems to indicate your home directory. This means you can easily return to your home directory by typing `cd ~` directory. `cd -` is another shortcut that will take you to the directory you were in before the current one.

Sharing

A very useful thing to do is to archive and/or share your keys. A common way of doing this is to export them as an ASCII armoured version. What this means is that the key is encoded in a form that is printable, rather than the binary format it's commonly stored in. This makes this a much more convenient form for printing out (and keeping safe) or for sharing with others via email, or even web pages. You can achieve this using the following command which exports the public component of your key:

```
gpg --output mykey.asc --export -a keyid.
```

You should replace *keyid* with the hexadecimal representation of your key above³. The output of this should look along the lines of:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQGNBGMQvGgBDAC1uEK7O5oFC8NE8A2dXog/1FKPoQcHoUrh8k6e8x+AN6dAPCMg
XULUberLVWpP2n1Q1eVGYbShPAwT7QyB4zyEvkflU5e2YBRZbmSw+fBXJfHERW9H
...
96FlccWitD9WUj3cx5dQLihG7ttbxC2CmfxxqdnM5aBFyqG94Xtun8H/PbED4x0C
Xli8/TRPEMI+Z5ma9A==
=9nQx
-----END PGP PUBLIC KEY BLOCK-----
```

This is your public key file you can share via email, or upload to servers. At the end of this your `.gnupg` directory should look like:

```
cnd@cnd22:~/gnupg$ ls -la
total 32
drwx----- 4 cnd cnd 4096 Sep  1 14:09 .
drwxr-x--- 5 cnd cnd 4096 Sep  1 13:54 ..
-rw-rw-r-- 1 cnd cnd 2505 Sep  1 14:09 mykey.asc
drwx----- 2 cnd cnd 4096 Sep  1 14:06 openpgp-revocs.d
drwx----- 2 cnd cnd 4096 Sep  1 14:06 private-keys-v1.d
-rw-rw-r-- 1 cnd cnd 2011 Sep  1 14:06 pubring.kbx
-rw----- 1 cnd cnd   32 Sep  1 13:54 pubring.kbx~
-rw----- 1 cnd cnd 1240 Sep  1 14:06 trustdb.gpg
```

Signing

Signing a file attests that you either trust it or are vouching for its contents. Create a text document called *First.Lastname.txt*⁴. In this file add a brief description of who you are and include your ssh public key you previously generated. The id on the key should match your `@stud.noroff.no` email address. This is typically similar to what you would do when sending someone your ssh key to add to a server for access. Now that you have created the file you need to sign it with your PGP key. There are two ways to do this:

1. Signing with an embedded signature `gpg --clearsign file`. This generates a file called *file.asc* which has a structure similar to the below. This contains the message protected by a ASCII Armour block containing the digital signature.

```
-----BEGIN PGP SIGNED MESSAGE-----
```

```
Hash: SHA512
```

```
<message>
```

```
-----BEGIN PGP SIGNATURE-----
```

```
iQGzBAEBCgAdFiEEDlfKKwrbliKr6Yxb5t1dqsrtzq8FAl+PDhkACgkQ5t1dqsrt
zq+00Qv+KK3+qwEy7bv/PvXybs2W6UMTP9gmGjoFUxAtTQM/PQXT6YUqqi99kUEt
Ter3byyqGFDaVNBnqBh9fgksnprStViGvi/BfPmr1FGdHFW1g+yrtUCM/+9T6iwY
```

³in my case this is EFD62... C1F21D

⁴Replace these obviously :-)

```
qfKTqqIErntbNVd3SXV/4g0y5xJb//ryK7zFUTPcm5pj9hbWqHVW1sW0HpIcMWI2
jtqojNY9jqp3xq9B70Xa01Vq4Nlzx0Erw47QNUFhaIiEhY/gC35mBewA5WLJqJug
x6J+Z/FJg5UahBh1kCBq1SmoA7T0biS2jIVahfZTHRgIN2C0SedGIIoqPK0Z1LgN
IkcqRntYuWgKJzBhovA5VgwHw627ckoT2xdP7KdZF0UXQPF2ndaGKdxLCMH4ku6C
s84mViJ8EOGXnHfmLFH7TdV1pprJ1L+XjV0+2XDPYLBc3d0c5degMgK1x+HV7073
oKle8AqonStGcfNuGmUn5EmWBmo5hWZ00M7FbR9HZHMopgU6jwNu3sh3FevQBU4j
7u08twf8
=NToY
-----END PGP SIGNATURE-----
```

This is the approach you would take when sending an email that is signed.

2. The second way is to create a standalone digital signature file, that can be used to verify another file. This is typically used to accompany digital files or larger files such as ISO's. `gpg --sign cnd.text.txt` creates *file.gpg*. This is a binary file so cannot be easily printed.



Commandline Arguments

gpg has a number of commandline arguments which control its operation. In particular it makes use of what are considered 'modern' more descriptive arguments (such as `--version`). What is **very important** to notice about these is they use **two** dashes preceding the argument. missing one of these can sometimes lead to errors which are not immediately apparent as to their cause. There are some single character argument/parameters that only use a single leading dash. See `man gpg` and `gpg --help` for more details.

Putting it to work



Now that you have the basics worked out, you need to use `gpg` to accomplish the following tasks (additional resources are in the Readings section below, but you should read the supporting manual pages as well).

1. Create a text file containing your name, Noroff email address and SSH public key and sign it (with ASCII armour mode - ie the signature is embedded in the text file)
2. Verify the signature
3. Make a copy of this file and make a change (e.g. add a space). Re do verification. Did it pass ?
4. Post the working text file to the PGP activity channel on Teams, so others in the class can see it
5. Export your PGP/gpg **public** key, making sure it contains your Noroff email address
6. Exchange keys with other members of the class and ensure you can encrypt and decrypt each others files.
7. Use these imported public keys to verify signed files that have been posted on Teams in the PGP activity channel.
8. Use **your** public key to encrypt a file for **yourself** (for example your SSH private key or a simple text file). Verify you can decrypt it. This is possible because you are using the corresponding **private** key to that you encrypted it with.
9. Create a brief text message, and encrypt it for someone else in your class. Send the text via email (either pasted into the body or as an attachment) - What impact does the differing encryption approaches have ?
10. Receive and decrypt a file you have been sent.
11. How do you verify that the sender was who they claimed to be ?
12. Repeat steps 10-12, but now include at least two recipients for the encrypted file. How does the size change ?
13. What benefit do you think comes from not having to encrypt files for multiple recipients individually ? What are the drawbacks ?
14. **Upload your exported key to the submission box on Moodle (under Week 2's tutorial).** These will be loaded into a master keyring on Wednesday 6th September at 12h00 that **will** be used later in the course.
You only have to upload a copy of your **public key**. This should **not** be signed (as it cannot be verified without having the key loaded).
15. **It is your responsibility to ensure that you keep a copy of your private key! Make a backup now on a system different to where you generated it/have it stored.**

Now that you have completed these tasks with `gpg`, you should find repeating them with a graphical tool such as GPG4win much simpler. For Linux and MacOS there are a number of alternates you can look at as well.

**Skilled up ?**

Before proceeding ensure you can do the following:

1. Encrypt a file
2. Decrypt a file
3. Sign a file
4. Verify a signed file
5. Import a public key you have received
6. Export and share your **public key**.

Remember to upload your **public key** on Moodle



The following online resources are a good place to start, and some contain further tutorial examples. Be aware that some refer to older versions of Ubuntu, but the `gpg` command line interface has been fairly stable for a number of years.

- [UBUNTU:GnuPrivacyGuardHowto](#)
- [How To Use GPG to Encrypt and Sign Messages on an Ubuntu VPS](#)
- [How to Generate and Manage GPG Keys on Ubuntu](#)
- [GPG Tutorial](#)
- [ENCRYPTING EMAIL IN OFFICE 365 WITH PGP](#)
- [Making and verifying signatures](#)

2 LAMP Stack Setup

Later in the course we will be looking at securing web facing platforms. As you are well aware a web based system consists of significantly more than just the 'web page' that is typically seen by an end user. While at its simplest form it could just be a static HTML page served up by a web server, most applications require the more comprehensive 'full stack' of a web server, application server (something to produce dynamic content such as a PHP processor), database server and the underlying operating system(s).

One of the most popular of these stacks is the LAMP stack – Linux, Apache, Mysql, & PHP. While each of these components can be swapped out (for example BAMP using BSD family Operating Systems as a base, and keeping the same others elements), the principles remain similar. MariaDB and PostgreSQL are to popular alternate RDBMS systems that are swapped into the stack.

Given the popularity of this software stack, it serves as a good basis with which to explore the defensive side of Computer Network Operations (CNO).

This task is very much preparatory work for tasks in the coming weeks. It is worth doing now, as it will be assumed to be complete with future tasks and thus no time allowance will be made for it. You need to ensure the following components are installed on your server, and operating correctly.

- Apache
- Mysql⁵
- PHP

These can be installed using the apt command to install appropriate packages. The default configurations that they are packaged with the software should provide a working base configuration as a starting point. You may need to restart the system to ensure all services are running.

The following are resources that provide more detail on the process. Read though them and see which one makes the most sense to you. Bear in mind that some refer to VPS or cloud hosted systems. Substitute your own server as needed. While we are using a newer version of the base OS there are no substantive changes in the way it functions and so the articles below remain relevant.

- [How to Install LAMP Stack with PhpMyAdmin in Ubuntu 20.04](#)
- [How to Install LAMP Stack on Ubuntu 20.04 Server/Desktop](#)
- [How to setup LAMP server on Ubuntu 20.04 Focal Fossa](#)
- [Install Apache, MySQL and PHP \(LAMP\) on Ubuntu 20.04 LTS](#)



PHPMyadmin

PHPMyadmin is **not** something one would realistically install on a production/internet facing server. However it is included in many guides, and you may find it an easier way to interface with the database. be aware that if not properly locked down, it can render your server vulnerable.



Notekeeping

Keep notes as to where you deviate from any standard configurations. These are really useful when building systems in the future, or for building your own packages or scripts that can be use to automate future deployments. Probably the most important reason to do this is so you know what you have done when something goes wrong and can go back and review prior work.

⁵ **MariaDB** is functionally identical to Mysql, but relates to a split form several years ago when Oracle took over control of the MYSQL core development, and members of the Open Source community saw this as a potential conflict of interest, and thus forked a new variant form the source at that point.

Verify the Install

Once your software components are installed, you need to verify that they components are working. The following are a very simple set of tests for you to perform to ensure your system is running as expected. Should things not work later, come back and verify that these all pass.

Apache



1. Is the web server software running ? Can you verify this in the process listing (`ps` and `top` can help)?
2. Is the server listening on port 80/tcp as you would expect? Are there any other ports? What would they be? (`netstat` is your goto here)?
3. Can you connect to the port using `nc` or `telnet`? Can you connect with a web browser?
4. Can you find the web server log files? There are two key types of logs – `ERROR` and `ACCESS`. Can you see your access attempts in the logs?
5. If things are not working what do the log files say?



Logs and Errors

The vast majority of logs on unix, and unix-like systems such as Linux, are kept as plain text which are appended to (lines added to the bottom). This makes them both human readable, and able to be searched/manipulated using standard tools. The command `tail -f` is a create way to follow the logs live as entries are being added. Running this in a separate window is a great way to debug or monitor services.

PHP

PHP is a little trickier to verify as the typical installation runs as a module inside Apache rather than as a separate process (as when using PHP in CGI⁶ mode)



1. Can you confirm the PHP module is activated in the `APACHE` configuration? *Note: Debian family operating systems have a slightly different approach to enabling this compared to other platforms, so be aware of this when you are looking at instructions you may have found elsewhere.*
2. Connect to the web server and and issue a manual request (or snoop the headers in your browser) and verify that Apache is reporting a PHP module being loaded.
3. Can you execute PHP code? The easiest way to do this is creating a simple script containing the following in a script called `test.php`:

```
<?php phpsysinfo() ?>
```

Place this in the webserver root directory, and then access it via your browser. You should be greeted with a dump of all the information from the built in PHP module.
4. Can you find the access in the web server log files? What about details of the module being loaded in the logs?
5. If things are not working what do the log files say?

⁶Common Gateway Interface

Mysql

MySQL is required here in order to be able to complete portions of later tutorials

The database in many ways forms the heard of modern applications, and are sadly an often neglected component in the overall security strategy.



1. Is the database software running?
2. Is it listening on the network port you would expect? What IP address(es) is it listening on?
3. Can you connect to the database using the client? Typically you would just run a command such as `mysql` as the system administrator. You will likely need to pass a parameter of `-u root` to this to tell it to connect as the root user when run as your normal user.
4. Can you list any databases that are in existence on the server?
5. Find the database server log file and look though it.



Using Mysql

When you make use of the `mysql` client, you enter a different command environment. Commands must be followed by a `;` in order to execute, for example:

```
mysql> show databses;
```

To exit this environment (use `\q` (quit) or press CTRL+D.



SNAP, you are safe!

You can (should) use the Snapshots feature in VMware to take a snapshot of your server before you start. If you mess up this proves an easy way to 'roll-back' to where you were without having to re-install. This is one of the best features for using Virtual machines for learning environments!



Unix Sockets

Mysql offers to modes of running (as defined in its configuration file)⁷, One of these is using a local socket. This is a special type of file more commonly known as a Unix Socket. Using this allows for communication by clients to the `mysqld` process without using a network. This means the service is not exposed on 3306/tcp. The effect of this mode is that only local applications can connect to the database, no remote clients or systems. You can verify this by looking for listening TCP sockets on 3306 using tools such as `netstat`. If the database needs to be accessible over the network this mode will not work, but for our purposes at this this is fine. The default configuration has `mysql` listening only via the loopback address.

⁷/etc/mysql/mysql.conf.d/mysqld.cnf

3 WordPress

As a test case for securing existing applications, we will be installing [WordPress](#)(⁸). The purpose behind this is to demonstrate defence in depth where we can add layers of protection to an application, without having to modify the application itself - as is the case for many closed-source offerings you may encounter.



Whats in a Name?

You will have been accessing your VM server via its IP address. While this is workable, in the absence of a DNS name its not ideal. As we deal with more web related activity, it is much easier to reference by name. There are a number of approaches to this in (in decreasing order of difficulty):

1. Set up and operate your own domain and DNS
2. Use a free/paid cloud DNS that will resolve a name to the IP of your VM
3. Fake it! We dont need no DNS!

In order to do the last, you can edit what is known as the `hosts`⁸ file on your client (and ideally server so they have a common view on what they are called.). On a Linux system you edit `\etc\hosts`. Instructions can be found in the two links below that cover MacOS and windows as well.

- [Editing Windows hosts file @wikiversity](#)
- [How to Edit Your Hosts File on Windows, Mac, or Linux](#)

After completing this, you should be able to access your VM by name. This can make the installation of services such as WordPress much easier.

Installation



Using the guides below, and building on your existing LAMP installation, you need to get WordPress installed and configured to the point where you can access the main page. There are two differing approaches to this:

1. Install the Ubuntu packaged version
2. Install WordPress by hand using a downloaded `tar.gz` file

Each of these has its own benefits and drawbacks. Do some reading though the guides and other resources before diving in. Make sure you keep a record of what you did. Some suggested starting points for your reading are:

- [How To Install WordPress on Ubuntu 20.04 with a LAMP Stack](#)
- [How to Install WordPress with Apache in Ubuntu 20.04](#)
- [Ubuntu 20.04 Wordpress with Apache installation](#)



It is suggested you install your WordPress instance in `/blog/` for simplicity.

⁸The hosts file goes back to the early days of the Internet before DNS. Each machine had a list of all other hosts on the network consisting of their name and IP address. You can find out more on the history and operation of this at [Wikipedia](#)

Testing



Once you have WordPress installed, ensure the following:

1. You can get to the blog landing page
2. You can post an entry
3. You can upload a file and link to it
4. You can upload an image and link to it in a blog entry
5. You have at least one blog entry
6. You create a second non privileged user (i.e. can only post comments not new entries)



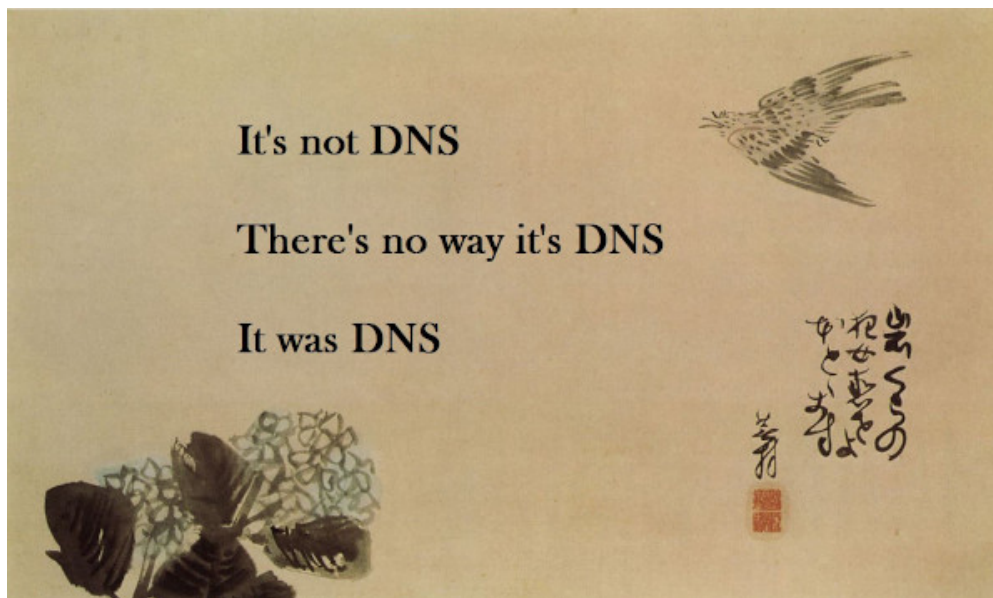
Taken a snapshot?

You now have your baseline system setup. It would be a good idea to save a snapshot so that if any further configuration goes wrong you can return to a known good state.



Remember to update the entry in your hosts file if the VM IP changes - failing to do so can lead to hours of frustration when things stop working.

In light of the above, some sage words of wisdom around the importance of DNS as conveyed in a Haiku⁹.
With thanks to [@nixcraft](#).



⁹A [Haiku](#) is an ancient Japanese form of poetry. This has become a popular form of writing in [English](#) too.

4 Multi Factor Authentication

As discussed in the lecture, adding a form of authentication to replace, or in addition to a traditional password can make a significant difference to mitigating a number of attack vectors. This task implements Time-Based One-Time Password Algorithm (TOTP) Protocols ([RFC6238](#)) to allow login to our server using a time synchronised token rather than (or in addition to) a password, or ssh key. These are similar to the earlier HMAC-Based One-Time Password Algorithm (HOTP) ([RFC4226](#)).











Do not get tied up in the actual algorithm and how it works, we are using an implementation of this. The [RFC](#) and [Wikipedia](#) provide much more detail and explanations if you really wish.

This can then be used by any application applications that implement the TOTP protocol.



You will need to download and install one of the following apps for your mobile device. These are suggestions, but there are other alternatives - including the OneProtect client¹⁰

- Google Authenticator [Apple Appstore](#)  [Google Play](#) 
- LastPass Authenticator [Apple Appstore](#)  [Google Play](#) 
- Microsoft Authenticator [Apple Appstore](#)  [Google Play](#) 
- FreeOTP¹¹ [Apple Appstore](#)  [Google Play](#) 



Take care

There is a strong chance of you locking yourself out of your system via SSH during this setup. You can recover access by logging in on the VM console. Before you start make a copy of the `sshd_config` file so that you can easily copy it back if there is a problem.

Setup



Configuring sshd

The SSH daemon (server) is configured using `/etc/ssh/sshd_config`. Line 12 of the default version of this file includes additional configurations from `/etc/ssh/sshd_config.d/*.conf`. SSHd works quite differently from many other systems in that the first time an option is defined this is what is used. To make your life easier it is **strongly suggested** that you do the following¹²:

1. Edit `/etc/ssh/sshd_config.d/50-cloud-init.conf` and comment out the single line using a hash character #.
2. Edit `/etc/ssh/sshd_config` and uncomment line 57:
`#PasswordAuthentication yes`
3. Save the file, and test it using `ssh -t` as root. You can use `-T` if you want more detail, including a dump of what all configuration options are set to.
4. restart the SSH service using:
`sudo service sshd restart`

Undertaking this **before** continuing with any further modification will make your life easier.

¹⁰There have been enrolment issues with this previously - your mileage may vary

¹¹<https://freeotp.github.io/>

¹²this may not be considered good practice in the real world but does simplify things for our purposes



A number of guides are available relating to the configuration and setup needed to get this working. The following are recommended as a selection for you to look at.

Make sure you have your backup user already configured and working **before attempting this!**

- [How to Use Two-Factor Authentication with Ubuntu 20.04](#)
- [How to use Google Two-Factor Authentication with Ubuntu 22.04](#)
- [Set Up SSH Two-Factor Authentication \(2FA\) on Ubuntu Server](#)
- [Configure SSH to use two-factor authentication](#)



Backups

When editing system files its good practice to keep a backup of files unless you know what you are doing. Its better to comment out existing entries and add notes as to that you are doing (and where you found information) rather than just changing or deleting entries. This makes it easier to change back to previous configurations when you are done testing, or need to undo changes you have made.



Impact Consideration

Once you have your login working, consider the following:

- How does this compare to simple passwords in terms of security offered?
- How does this compare to the previous task of using a cryptographic key to log into the system?
- Where do you think that this TOTP approach could be preferable to using a cryptographic key?
- What do you think a downside of this approach would be?
- How could you deal with a lost/stolen/damaged mobile authenticator?



`man sshd_config` provides a lot of detail around the various options that you can use. You should pay attention to the `Match` directive.



Extending Function

Think about how you could extend this functionality. Consider trying the following:

- Can you enable this for some users?
- Can you enforce this for certain users or groups
- Can you require this if a user logs in from certain IP ranges?
- Why might you want to do this ? What would be the impact be of inverting the above ?
- Can you disable this if a user logs in from a known trusted IP range?
This may be particularly useful once you know it works to not require your primary user to have to enter a TOTP token every time you login from your client system.

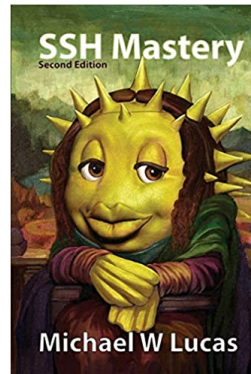
To accomplish these one needs to spend some time looking at PAM documentation, and specifically some of the configuration options in `sshd_config`.

**Rolling it back**

While 1FA is a strong form of authentication, it can also be quite comber-some to use constantly, particularly in a test environment such as we are using. When you have completed the task, you are encouraged to keep a copy of the working config files for 2FA, and then disable the 2FA prompts so you do not need to use it every time you log in.



For those that are interested, Michael Lucas' book *SSH Mastery: OpenSSH, PuTTY, Tunnels and Keys* provides a excellent resource for advanced OpenSSH configuration:



5 Text Processing: awk, sed, grep, and friends

If one considers BASH as the interface and glue in working on a unix system, the utilities are what actually help you get work done. the vast majority of data that we deal with is text (or can be converted to text). This is useful as it means that its human readable (and hopefully understandable) Fortunately several tools are available as part of a standard unix distribution that provide a great degree of flexibility when dealing with text data are `awk` and `sed`. These tools together with `grep` significantly extend the type of processing that can be done using bash scripts (which really are glue holding strings of other tools/commands together). The ability to use these commands is important where you cannot use a more full featured language such as Python or Perl often due to security restrictions. Alternately when large datasets are being processed, more traditional tools such as spreadsheets generally do not cope as they try hold all the information in memory at once, rather than processing line by line.

The operations presented here are a highlight of some of the most relevant features for these commands that you may need in this course. There is a wealth of information not covered. You are encouraged to look at the command manual pages. You should also refer to the linked additional information in each section. These contain additional exercises and examples which you should ensure you understand.



You will need `Tut2-2023.zip` which contains the files needed for the activities in this Task. These can be found in the `textproc` directory.

Checksum:

SHA256: `0b63057e7bc7c2b443db4ff00a2a7f7b13a8d44a4866015eb4e18a0e8540c92d`

Be sure to validate your download is correct.

The password for the file was given in this week's lecture.

awk

`awk` is a scripting language used for manipulating data and generating reports. The `awk` command programming language requires no compiling and allows the user to use variables, numeric functions, string functions, and logical operators. While very flexible, there are a few key features commonly used, and of specific value to processing data in this course. The general syntax for the `awk` command is:

```
awk options 'match {action}' input-file > output-file
```

These can be broken down as:

options The only one needed for our purposes is `-F fs` which defines what to use as a record separator. By default `fs` is set to whitespace, but can be set to `:` or `,` to define how to split records.

match This is seldom used, but you can include matching and regular expressions here.

script Is where the 'work' of the program happens. The most common action used is `print` which is followed by a number of variables denoted as `$1`, `$2` etc. These represent the fields.

input-file File on which the actions should be performed, or `stdin` if omitted.

output-file File in which output should be saved (as indicated by the redirect character `>`). `awk` prints to `stdout` so can easily be chained with other commands.

Worked Example

This is based on the file `task3mini.csv` obtainable from Moodle. A brief look at the file shows it is a csv and has a number of fields separated by commas (,) – a very popular means of packaging information that can be processed by a wide variety of tools, from spreadsheets to programming languages, and now at the command line!

```
$ head task3mini.csv
"host","domain","class","subclass"
"000directory.com.ar","000directory","legit","legit"
"000webhost.com","000webhost","legit","legit"
"001fans.com","001fans","legit","legit"
"01-telecharger.com","01-telecharger","legit","legit"
```

```
"010shangpu.com","010shangpu","legit","legit"  
"011info.com","011info","legit","legit"  
"0126wyt.com","0126wyt","legit","legit"  
"012global.com","012global","legit","legit"  
"01basma.com","01basma","legit","legit"
```

If we wanted to print out the domain column only we could use the following:

```
awk -F, ' {print $2 }' task3mini.csv
```

This command will read in the file `task3mini.csv` and split it using a comma (,) as a record separator. The *action* is to print the second field as denoted by `$2`. Output should look similar to the following:

```
$ awk -F, '{print $2}' task3mini.csv  
"domain"  
"000directory"  
"000webhost"  
"001fans"  
"01-telecharger"  
...  
"0755car"  
"0769auto"  
"0791quanquan"  
"0800-horoscope"  
"0996tuan"  
"0n-line"  
"1-resources"
```

While this provides a simple example, there are a number of issues with the output. Fortunately the next two tools can help clean up the output. This is possible using just `awk`, but for many people getting to grips with the system, breaking up the process is both easier to understand and far less error prone.



Tips for using `awk`

Some quick tips for using `awk`

- Make use of the `-F` parameter where it makes sense, splitting on a value can make the rest of your life easier. Value can be any character. Remember that some may need escaping (such as the pipe `|`).
- Remember that the *action* component passed to `awk` **must** be enclosed in **both** single quotes (') and braces ({ }).
- Remember the KISS principle. If things are getting very complicated, maybe its time to rethink your approach.
- A good approach for debugging is to remove complexity until you get back to expected output. Are you doing this in the simplest way possible ?



`awk` Resources

Some good resources for learning more about `awk` can be found at:

- [Awk Tutorial](#)
- [Introduction to AWK](#)
- [Simple Awk](#)

sed

`sed` is a powerhouse tool on the Unix command line, providing inline editing features such as search and replace, text matching and re-ordering, and general regular expression processing. The name `sed` is derived from the function of the tool as stream editor. This means it is able to perform edits (and thus general processing) on text as it passes through the tool, line-by-line. It can also be used for modifying files automatically. The primary use case for `sed` is as a filter that processes inputs and outputs the changed text, based on the instruction that are provided to the command.



REGEX Beware

Many people often fall into the trap of jumping directly to regular expressions. While extremely powerful, the old adage of "*you have a problem, and use regex, now you have two!*" has more than a grain of truth.

The general syntax for the `sed` command is:

```
sed options 'script' input-file > output-file
```

These can be broken down as:

options While there are a range of options that can be used, the most important needed for our purposes is `-e`. This defined that what follows is a script to be run, and allows for the chaining of multiple actions in a single line.

action Is where the 'work' of the program happens. Likely the most commonly used function is substitution or `s/` function. This provides a capability for search and replace of strings in text. This same functionality can be used with regular expressions to allow for far more selective and powerful operation.

input-file File on which the actions should be performed, or `stdin` if omitted.

output-file File in which output should be saved (as indicated by the redirect character `>`). `awk` prints to `stdout` so can easily be chained with other commands.

The use of these components is shown in the worked example below. You are strongly encouraged to refer to the system man page for the command.



Search Delimiter

Typically the majority of cases, and most examples you will find use the `/` or slash as the delimiter in the expression. You can however use a wide range of matching characters. This is very useful when you are wanting a slash to be something you match against. Commonly used characters for delimitation are: `|`, `,`, `_` and `:`

Worked Example

This is based on the file `task3mini.csv` obtainable from Moodle. A brief look at the file shows it is a csv and has a number of fields separated by commas (`,`), as we saw with `awk` above.

We can try clean the file up to remove the `"` characters around each field.

```
$ sed -e 's/"//g' task3mini.csv
host,domain,class,subclass
000directory.com.ar,000directory,legit,legit
000webhost.com,000webhost,legit,legit
001fans.com,001fans,legit,legit
...
0n-line.tv,0n-line,legit,legit
1-resources.com,1-resources,legit,legit
```

Using a more complex approach and changing commands lets replace the value legit at the end of the line with LEGIT. Note the use of the \$ to anchor the match expression to the end of line. If this is not familiar, you may need to go take a refresher on regular expressions.

```
$ sed -e 's/"//g' -e 's/legit$/LEGIT/g' task3mini.csv
host, domain, class, subclass
000directory.com.ar, 000directory, legit, LEGIT
000webhost.com, 000webhost, legit, LEGIT
...
On-line.tv, On-line, legit, LEGIT
1-resources.com, 1-resources, legit, LEGIT
```

This type of search and replace action is what is by far the most frequent use for sed.

**s/syntax tl;dr**

The basic syntax for the substitution command is as follows:

s/pattern/replacement/action

pattern what you would like to match either plaintext or regular expression. pay attention to escaping special characters and the need to use regular expression anchors of ^ and \$ as appropriate.

replacement text to be used to replace the pattern. This can be empty to delete the pattern, can be plaintext, or can make use of Regular Expression positional matches from the match pattern.

action is almost always safe to use **g** for a global replacement.

`sed -e 's/fun/sun/g'` replaces *fun* with *sun*

grep

grep (and its variants egrep and fgrep) are used to search a file (or *stdin*) for a particular pattern of characters. This pattern can be simple strings or more complex regular expressions. The general syntax for the grep command is:

```
grep options pattern input-file(s) > output-file
```

These can be broken down as:

options The most important needed for our purposes are:

- i Ignores, case for matching
- E Treats pattern as an extended regular expression
- w Match whole word
- o Print only the matched *parts* of a matching line, with each such part on a separate output line.
- v Inverts the match and prints lines that **do not** match the pattern

pattern Is what is matched against. This can be a simple string such as `hello` or more complex regular expressions such as `"^hello|bye$"`

input-file(s) File(s) on which the actions should be performed, or *stdin* if omitted.

output-file File in which output should be saved (as indicated by the redirect character `>`). `grep` prints to *stdout* so can easily be chained with other commands.

The use of these components is shown in the worked example below. You are strongly encouraged to refer to the system man page for the command.

Worked Example

This is based on the file `task3mini.csv` obtainable from Moodle. A brief look at the file shows it is a csv and has a number of fields separated by commas (`,`), as we saw above.

Using the `-v` to look for lines that do not contain the word `legit`. we see that we are left with just the header row from the file.

```
$ grep -v legit task3mini.csv
"host","domain","class","subclass"
```

Looking for domains or hosts containing the term `car` or `auto`, we essentially match for entries containing either of these.

```
$ grep -E "car|auto" task3mini.csv
"0755car.com","0755car","legit","legit"
"0769auto.com","0769auto","legit","legit"
```

Note the difference when the `-o` option is used. Why are the entries printed twice?

```
grep -oE "car|auto" task3mini.csv
car
car
auto
auto
```

Other useful tools

The unix command line offers a myriad of useful tools that can be chained or 'plumbed' together. Most tools are built on the philosophy of 'one tool one job'. Some of the more common you may find useful are

wc provides a count of words (-w by default, or lines when using -l . It can also be used to count the number of bytes in the file using -c.

sort sorts input data. Several options exist for how this is done. The two most useful are -n and -r . See the man page for more details.

uniq removes duplicates from input. The -c parameter returns a count of the number of observed instances of each entry. This can be thought of as similar to a crude 'GROUP BY' clause in SQL. **This command expects data to be sorted.**

cut trims columns of characters, and can be very useful for removing parts of lines that are not of interest, but relies on absolute position.

head takes *n* lines from the start of the file. Default is 10.

tail takes *n* lines from the end of the file. Default is 10.

cat concatenates a file, use this to write a file to screen or to pass to other commands with pipes. the **tac** command does the same but prints the file out in reverse.

tr translate, squeeze, and/or delete characters from standard input, writing to standard output. Useful for changing case, stripping DOS mode files and other text manipulation.

Worked Examples

```
# sort the file
$ sort task3mini.csv | head -5
"000directory.com.ar","000directory","legit","legit"
"000webhost.com","000webhost","legit","legit"
"001fans.com","001fans","legit","legit"
"010shangpu.com","010shangpu","legit","legit"
"011info.com","011info","legit","legit"
# Run wc
$ wc task3mini.csv
  30   30 1296 task3mini.csv
#now only count lines in the file
$ wc -l task3mini.csv
30 task3mini.csv
#grab the first character in each line
$ head -5 task3mini.csv | cut -c1
"
"
"
"
"
# Print from the 2nd character of a line onward
$ head -5 task3mini.csv | cut -c2-
host","domain","class","subclass"
000directory.com.ar","000directory","legit","legit"
000webhost.com","000webhost","legit","legit"
001fans.com","001fans","legit","legit"
01-telecharger.com","01-telecharger","legit","legit"
```



Grabbing fields

The cut command can be quite useful, but many people find it easier to explicitly grab fields that you are interested using something a little more precise such as awk. Experiment and build up your own cheat-sheet of commands and snippets.

Bringing it together

The commands above can be used in combination to process and analyse the file. The following command string determines how many times the term *legit* occurs in the *class* column, and cleans up the output.

```
$ tail -29 task3mini.csv | sed -e 's/"//g' | awk -F, '{print $3}' | grep -i LEGIT | sort | uniq -c
29 legit
```



1. What do you think the purpose is of the `tail -29` command at the start, and what is an alternate way(s) of doing this?
2. What do you think is a better way of processing the data, especially if we are going to be doing a number of queries on it?
3. Work through all the examples above and make sure you understand them.
4. Try out the different command line options for the commands. Refer to their manual page for details.
5. There are a significant number of online tutorials for these command which you can refer to for more examples. One of the best ways to learn is through trying to solve problems.



Break it down

When working with data it's worth building up your solution step by step. It's always useful rather than trying to build one complex pipeline to have each command take input and output to a file which can be used by the next step. This helps greatly when debugging, and when you are confident it is working you can merge the commands into a single pipeline.



Additional Resources

Some additional resources you may find useful for these tools.

AWK

- [AWK command in Unix/Linux with examples](#)
- [Awk Tutorial](#) be sure you understand the [Basic Examples](#).

SED

- Sed Command in Linux/Unix with examples [Part 1](#) and [Part 2](#)
- [Sed - An Introduction and Tutorial](#) by Bruce Barnett. This also links to an advanced [Sed Reference chart](#).
- [A sed tutorial and reference](#) by Alex Harvey
- [Unix sed Command Tutorial with Examples](#) - medium.com has limited free views¹³.

Others

- [grep command in Unix/Linux](#)
- [cut command in Linux with example](#)

¹³Edit your cookies or use incognito mode

Real World Problem Solving

For this exercise use the file `legit-dga_domains.csv` found in the zip. The format is identical to the file used in the example above which is a small extract of it. This is a file containing domains and a high level classification of “dga” or “legit” along with a subclass of either “*legit*”, “*cryptolocker*”, “*goz*” or “*newgoz*”. This data was produced by Jay Jacobs ([@jayjacobs](#)) for his book Data-Driven Security. You can read more about the background to this file in his blog article [Building a DGA Classifier](#).



Building on the examples above, use `grep`, `sed`, `awk`, and other command line tools to undertake the following:

1. How many lines does the file have?
2. What are the 6th to 16th characters of its md5 checksum?
3. Clean up the file for further processing removing the header and the ” characters. save the file as `cleandata.csv`.
4. What is its md5 checksum now?
5. How many different classes are used and how many entries in each class?
6. You notice some odd output from the above, due to some odd data. What is this data?
NOTE: One always needs to be wary of data that is not quite as clean as it seems.
7. How many subclasses exist within the *dga* category and what are the number of records labelled with these.
8. Taking the output above, print it as the category and then the count.
9. For the domains linked to cryptolocker, what are the top 5 (by count) of characters they start with?
10. For the host linked to newgoz, what are the TLDs (the part after the final . in the name) that are used? Print them in ascending order of count.
11. Which country code or ccTLD ranks highest within the goz classified domains?
12. What line number/position is Iran in the listing above? Alternate question:
What line number/position is Iran for domains classified as *legit*?
13. As above, but print only the Statement as:
`ir is in position x`
14. Print out the two letter country codes from question 11 capitalised.
*HINT: the **tr** command is useful for translating character mappings.*

NOTE: A set of answers above will be released mid-week for you to verify against. In the interim confer with your class mates. In terms of solutions there are many ways to achieve the right answer.



Validate your output While it is convenient to allow for automated processing of your data, you should be aware that real-world data is not always perfect or clean. Often this shows up in unexpected results or errors in processing the data that you would not expect. Take time to look at the data you are working with, as well as the outputs at each stage of the analysis.

**sed is hard!**

sed offers a very rich set of tools and functionality. the trick to using it effectively is to learn these as you need them, for the vast majority of cases sed is most useful when doing substitutions – the **s** command. Even within this there are a number of advanced features, most useful in a few edge cases. You need to be comfortable with the search and replace or substitution notation. The other functionality can be looked up when needed.

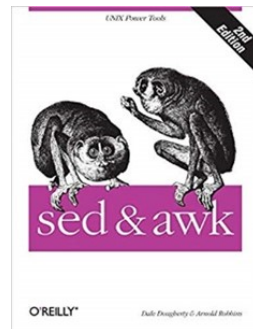
The same applies with many commands. Keep notes as a reminder, and commonly used phrases will soon become easier to remember.



Exercise your skills working with some text files (combining with grep is also useful). CSV file manipulation is another area to practice. Can you change the , in a csv file to a ; or replace all cases of " with ' ?



The use of sed and awk is important where you cannot use a more full featured language such as Python or Perl often due to security restrictions. One of the more authoritative resources for sed (and awk) is the book by Dale Dougherty & Arnold Robbins - *sed & awk* Second Edition.



6 Deliverable

At the end of this practical sheet you should be able to use your skills to solve the problem below. you must record the commands that you use to solve this in a text document. You should solve this preferably using command line solutions, although python scripts can be used for some parts. If used they must be included in the text document. you will be required to submit the text document (or portions thereof) in the *Engagement task* in week 3.

Introduction

Norwegian Networking

Cyber Security in general deals with a lot of network related data, and the problem you are facing is no different. You have been provided with a list of network blocks associated with Norwegian companies, by a Senior Threat Intelligence Analyst. These have been obtained from the Global RIRs¹⁴. Your tasks as up and coming analyst is to do some data manipulation and then answer some questions around. Pay attention to any formatting specifications laid out. You are given the data to processing in an encrypted zipfile along with a Postit Note.



Data file

File: deliverable-week2.zip

SHA256:f701c091890fd5d290c5a53a1361c78874c044959b094dca762055312105bbbe

Password: puvonpvg!



Cannot access the data?

After failing to successfully extract the file, you recall the Senior mentioned he had used a simple *Caesar Cipher* to protect the password on the note. You can use a number of tools to solve this, or do it by hand. you can also solve this using the text processing skills you have learned in this tutorial.

The file as provided is in the following format:

```
ripenc|NO|ipv4|109.179.0.0|65536|20090921
ripenc|NO|ipv4|109.189.0.0|65536|20091009
ripenc|NO|ipv4|109.199.192.0|8192|20100121
```

The columns can be defined as:

```
RIR|CC|Type|Netblock Start|Allocation Size|Date
```

These fields are defined as follows:

RIR The Regional internet Registry who 'owns' the IP block based on IANA allocations

CC The ISO allocated two letter country code

Type What type of record this is. Only ipv4 records are of interest.

Netblock start The first allocation of the netblock, ie the network number.

Allocation The number of IPv4 addresses in the netblock

Date The date on which the allocation was made by the RIR

Any other fields can be ignored. You are to ensure that after cleaning, the file only contains ipv4 data relating to NORWAY (NO), with no duplicates.

You have been tasked to reformat the file as follows (using the three lines above as examples) and to ensure there are no duplicate entries.

```
109.179.0.0,65536,20090921,ripeenc
109.189.0.0,65536,20091009,ripeenc
109.199.192.0,8192,20100121,ripeenc
```

¹⁴RIS is a regional Internet Registry who is responsible for allocating IP address blocks on the public Internet

Questions

Once the data has been reformatted and cleaned, you should answer the questions below.

For each question keep a record of the answer as well as the command you needed to run to get it.

1. How many unique netblocks are there ?
2. How many RIR's had blocks attributable to Norway
3. Which years had the most, and the least blocks allocated ?
4. Print a list of the size of allocations ordered from largest to smallest in the format of:
allocationsize,count
5. How many IP addresses are allocated to Norway?
Hint: several means can be used to sum values up find one that works for you.
6. What percentage of the total allocation to Norway do the largest three (3) categories from question 4 amount to?
7. To be able to use the data in a filter list on a firewall you have been asked to transform it to the following format, using CIDR notation to indicate the network size rather than the count of IPs:

109.179.0.0/16
109.189.0.0/16
109.199.192.0/19
8. Provide a list of /16 netblocks with the count of the number of allocated network for Norway they each contain



Notes Notes Notes

Make sure you keep clear notes, both on how you undertook each step, and what the result was. This includes the initial cleanup done. you will need these to refer to later in the course.



Check your answers

You can verify your answers via a self evaluation on Moodle, which will be available from 09h00 Wednesday morning.



Week Checklist

Check you have completed what needs to be done for next week:

1. Completed server configuration
2. Uploaded your PGP key on Moodle
3. Completed Task 5 and kept your notes and answers
4. Completed the Deliverable task, and kept your answers and notes

A News

The following are new articles of interest in the last week. These are all relatively short, and you should get in the habit of staying aware of what is going on in the industry at large. The topics discussed here fall in scope for the engagement tasks.



News Week of 28 August 2023

MALWARE [2023 MID-YEAR CYBER SECURITY REPORT](#)

The Check Point Research (CPR) team points to a notable resurgence in cyber-attacks across the globe. The second quarter of the year alone the most significant surge in the last two years.

ATTACK [Criminals go full Viking on CloudNordic](#)

CloudNordic has told customers to consider all of their data lost following a ransomware infection that encrypted the large Danish cloud provider's servers and "paralyzed CloudNordic completely".

TOOLS [Scarabs colon-izing vulnerable servers](#)

Despite this tracking and our detailed analysis of Spacecolon's constituent tools, we cannot currently attribute its use to any known threat actor group. Therefore, we will call Spacecolon's operators CosmicBeetle to represent the link to "space" and "scarab".

ATTACK [Bard's Tale – how fake AI bots try to install malware](#)

An ad by "Google Bard AI", suggesting to download and try out the latest version of Google's legitimate AI tool "Bard". It seemed odd for an internet giant to be using the services of another provider and my suspicion was triggered.

MALWARE [Ransomware Roundup](#)

This edition of the Ransomware Roundup covers Trash Panda and a new minor variant of the NoCry ransomware.

B Reading & Listening

Required Reading for the week is a seminal article reviewing the concepts covered in lectures.



Idrus, S.Z.S., Cherrier, E., Rosenberger, C. and Schwartzmann, J.J., 2013. A review on authentication methods. Australian Journal of Basic and Applied Sciences, 7(5), pp.95-107.
[Available here](#)

Estimated read time is 30-35 minutes.



What are you listening to?

Podcasts have become an increasingly popular means of getting information, and are often easier to consume 'on the move' as opposed to reading material. What Infosec/Cybersec podcasts do you listen to? Post any good resources on Teams.

Preparation Task

Find and listen to an episode of a security podcast, and write up a brief (100-150 word) summary including the following points:

- What the main point was
- What you learned
- Why you chose this podcast and episode

In addition make sure you include:

- The URL
- The Episode Title

Save this as a text file and then exchange this with another member of the class (or more) and discuss.

You **will** need this text file for **Engagement Quiz 2**



Go Faster

Most podcast apps (or even YouTube) allow you to play back audio faster than it is recorded. This can speed up your time getting through material often without any loss of understanding. Find a speed that is comfortable for **you**.

Server Hardening

Computer Network Defence - 2023/24

Version 1.0 04/09/2023

Following on from ACCESS CONTROL, and the system configuration you have been working on, we focus on security functionality that the Operating system can provide. These security features come 'free' (even on commercial Operating Systems). While commonly used and understood by Systems Administrators they are sometimes a mystery to those with out operations experience. A system administrator that understands their platforms can be a very valuable ally to the security team in terms of getting fine grained control and function out of the platform without having to purchase third-party products. The initial process of applying these security controls is known as HARDENING. The Operating System provides a first layer of security and isolation. This will be explored further in coming weeks relating to APPLICATION STACK HARDENING.

Linking back to the lecture, specific mechanisms are considered starting with a brief introduction to Kernel tunables, and File System security features. Kernel tunables will be revisited in more detail at the point we focus on on network related material. Linking back to Access Control and Password Security, we consider how to set strong passwords - and discover what a pain this process can be for end users (which revisits the strength from length rather than complexity argument). The tutorial is largely constructed as a series of steps to be followed, with a reduced content relating to 'problem solving'. Your attention is directed to the various questions for you to consider as you work though the material.

The practical tasks this week consist of three distinct groupings:

SYSTEM SECURITY [Tasks 1-3] Explore some simple (and safe) Kernel and Operating System configuration. Task 4 extends the work on Access controls, looking at how password security can be improved (and enforced).

APPLICATION [Tasks 5 & 6] build on the text processing you undertook in the previous week's tutorial tasks. We look at unstructured data in the form of logs generated by authentication attempts. The final components (including the deliverable) relate to application of knowledge and in particular the skills you gained last week. You will be examining authentication logs for a server, and extracting certain types of information of interest to a security analyst. This will feed into some further defensive work in coming week.

READING There are five brief news articles to review, along with a reading and listening activity.

These broad areas can all be worked on independently, with no inter-dependency between the sections. The order in which you approach these three areas is up to you. You are strongly encouraged to switch between tasks if you get stuck or find your concentration drifting. The Deliverable comprises of 13 short tasks dealing with log processing. Ensure you keep notes of **both** your answers **and** how you arrived at them. You are expected to have the tasks in this tutorial completed prior to the next week's tutorial. The activities there in build on the concepts covered and skills developed in the preceding tasks, as well as your prior weeks.

Something that has been apparent in the last few weeks is that there have been a number of cases of unnecessary frustration due to not reading, gaps in foundational concepts (primarily file manipulation, virtual machines and networking concepts) and especially not reading error messages being produced.

At this point in your academic career, making backups of your work (especially as cyber students where the domain exists to protect information) should be near automatic. **The responsibility to ensure you have copies of important material rests with you.** You have large volumes of cloud storage available via OneDrive that you can make use of.

Your attention is drawn to the end of this document (Section D) which contains A 'cheatsheet' for some text processing tools that you may find useful.

This tut *may* be updated during the course of the week with additional support material, make sure you have the latest version as shown on Moodle.

Contents

1	Linux Kernel Hardening	1
2	FS hardening	4
3	FS Atributes	7
4	PAM Auditing	10
5	SSH Data	14
6	Deliverable	19
	Resources	20
A	News	21
B	Listening	22
C	FS Notes	23
D		27

CHANGELOG

 v1.0 - Initial Release

1 Linux Kernel Hardening

The operating system is the first place to look at securing our systems. At the heart of an Operating system is the system `kernel`



This task is to be done on the Ubuntu Server set up in week 1



Before proceeding make sure you have a snapshot of your system. Some of the changes to be made can have a potentially negative impact. Where you edit files, make sure to keep a back up of them.



Backups and Changes

As you make changes to systems a good habit to keep is making sure you can undo those changes. VM Snapshots are a fairly drastic way to do this. These however are not applicable when working on 'real' machines. The following are good things to keep in mind.

1. Keep backups of files. Before editing or changing configuration files make a copy. For the purposes of the tutorial system making a copy in the same directory is sufficient. Using a suffix like `.orig` is a good way of denoting that the file is as it was originally. for others you may want to use the classic `.bak` or even a description of the task. This provides you with a very easy means of rolling back changes or reverting back if things stop working. There are generally few configuration changes that will result in an unbootable system.
2. When making changes in configuration files rather than deleting what is there, comment the line out, duplicate it and make your changes on the new line. Its always good to add your own comments as to what the change should do, or where you found information on why a setting needed to be set. This approach works very well for when changes are relatively minor and allows you to switch between configurations when testing. When more significant changes are being made, a backup is **still** recommended.
3. Your notes are probably the single most valuable backup resource. Ideally these should **not** be kept on your VM system you are experimenting on. Cloud storage and web based editors are a fantastic way of ensuring safety. Your notes are what should help you recover in the event of major disaster.

Kernel Controls

The Linux kernel has a wide range of settings that can be adjusted at *run time* as opposed to *compile time*. These are done through the `sysctl` interface. You can learn more about this by typing `man sysctl`. You can think of these as various flags, variables or 'knobs' that can be used to adjust the kernel's modes of operation.

As you work through these, you may find it useful to look up the appropriate `sysctl` using the [SysCtl-Explorer](#) website. This provides a little bit of information as documented (largely) in the system source code via a much more friendly web interface. Many of the settings relate to tweaking the way the network stack operates. These will be addressed and we will revisit the tuning in coming weeks when we focus on networking and network security.

For running each of these commands you need to use the format `sysctl sysctlname` to see the current setting, followed by `sysctl -w sysctlname=newvalue` to set it.

An example can be seen below checking the setting the `kernel.dmesg_restrict` options.

Start by checking what the existing value and functionality is as an unprivileged user:

```
$ sysctl kernel.dmesg_restrict
kernel.dmesg_restrict = 1
```

Now we can try look at the in kernel log messages using the `dmesg` command.

```
$ dmesg
dmesg: read kernel buffer failed: Operation not permitted
```

This is not unexpected as we saw previously the value was set to 1, this in general means enabled where the `sysctl` (knob) has a binary value.

To set (change) a value you can use the `sysctl` command again this time with the `-w` (write) flag and provide a value to set the `sysctl` to. This must be run (using `sudo`) with elevated privilege - ie. as the root user:

```
$ sudo sysctl -w kernel.dmesg_restrict=0
kernel.dmesg_restrict = 0
```

We can now retry seeing if a non privileged user can read the kernel log.

```
$ dmesg | head
[ 0.000000] Linux version 5.15.0-47-generic (buildd@lcy02-amd64-060) (gcc (Ubuntu 11.2.0-19ubuntu1)
[ 0.000000] Command line: BOOT_IMAGE=/vmlinuz-5.15.0-47-generic root=/dev/mapper/ubuntu--vg-ubuntu--
[ 0.000000] KERNEL supported cpus:
[ 0.000000] Intel GenuineIntel
[ 0.000000] AMD AuthenticAMD
[ 0.000000] Hygon HygonGenuine
[ 0.000000] Centaur CentaurHauls
[ 0.000000] zhaoxin Shanghai
[ 0.000000] Disabled fast string operations
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'
...
```

We can see here that changing the `kernel.dmesg_restrict` value from 1 to 0 **removes** the restriction on non privileged access to the kernel message log. This aligns with what is available on the [SysCtl-Explorer](#) site entry for this value. The default that Ubuntu 22.04 ships with is considered more secure and is a change from the past. In production this should definitely be restricted.

For the purposes of the tutorial, it provides a very low risk tunable to demonstrate functionality.




- Why would this be something you want to run on a system?
- What would the advantage be?
- Think how the information provided could aid an attacker?

Using a combination of exploration, experimentation and using online resources complete the following table:

sysctl Value	Secure Function	Test process
kernel.dmesg_restrict	1	Restricts access to kernel logs
kernel.kptr_restrict		Run dmesg
kernel.unprivileged_bpf_disabled		
kernel.yama.ptrace_scope		
kernel.kexec_load_disabled		



For those wanting to answer this using \LaTeX , a template for the table of values is available [here](#) via  GIST

Boot Parameters

Not all kernel options can be tuned at runtime. Some need to be passed to the kernel when it boots. These can be passed to the kernel via the bootloader. This is either done manually at boot time, or via configuration of GRUB.

Further details on kernel configurable can be found in the [Linux Kernel Documentation](#)



GRUB is the boot loading system used, to load the Linux kernel and ultimately the operating system). Be very careful with any changes made using this as you can render your system unable to boot. Make sure you have a current snapshot before making any changes.

You are NOT required to do any changes to GRUB.

2 File System Hardening



This task is to be done on the Ubuntu Server previously configured



Before proceeding make sure you have a snapshot of your system. Some of the changes to be made can have a potentially negative impact. Where you edit files, make sure to keep a back up of them. One of the easiest ways of doing this is to copy them and add the suffix of .orig or .bak. These then remain available for you to compare to.

Filesystem Security

One advantage of the Unix approach of treating most things as a file, is that we can use a consistent interface across different underlying physical, logical and virtual storage.

The Linux file systems are *mounted* (i.e. joined into the main system file hierarchy) using the `mount` command, with `umount` used to remove them.

Different filesystems offer a range of features, relating to security, performance and functionality (such as compression). What we explore in this task is some of the common features that can be used to enhance the security of a system. By making use of these options one can go a long way towards frustrating (and ultimately slowing down) attackers. This functionality is already available on a system and comes at no extra cost. Making use of these features **does** however require careful planning and thought.



STOP Before carrying on, you must complete the setup of the test drive, file-system and files that you are going to use below. These can be found in Appendix C.

Now that we have a prepared file system you can try out three specific options that Linux offers to enforce security at the fundamental level of the file system.

No Execution

The `noexec` option¹ for a file-system can be passed via the `mount` command.

```
$ sudo mount -o noexec /dev/sdb1 /demo
$ mount | grep demo
/dev/sdb1 on /demo type ext4 (rw,noexec,relatime)
```

You can see that the new option has been added and is reported by the kernel. Change into the directory and do a directory listing. Then run `id` as your standard user, and then run each of the three demo files.



- What results do you see different to when you set the system up?
- What happens if you try run these commands as `root` either directly or via `sudo`?
- What happens when you try run your shell script?
- What scenarios could this be useful to have as a mount option?

Unmount your file system.

¹There are very large selection of mount options that you can use — in various combinations — to customise what you want to do with a file system. Some specific file systems have their own esoteric and specialised options. For details have a look at the output of `man mount`

No SUID

The `nosuid` option for a file-system can be passed via the `mount` command.

```
$ sudo mount -o nosuid /dev/sdb1 /demo
$ mount | grep sdb
/dev/sdb1 on /demo type ext4 (rw,nosuid,relatime)
```

You can see that the new option has been added and is reported by the kernel. Change into the directory and do a directory listing. Then run `id` as your standard user, and then run each of the three demo files.



- What results do you see different to when you set the system up?
- What happens if you try run these commands as `root` either directly or via `sudo`?
- What happens when you try run your shell script?
- Why do you think this is?
- What scenarios could this be useful to have as a mount option?

Unmount your file system.

Read Only

The `ro` option for a file-system can be passed via the `mount` command.

```
$ sudo mount -o ro /dev/sdb1 /demo
$ mount | grep sdb
/dev/sdb1 on /demo type ext4 (ro,relatime)
```

You can see that the new option has been added and is reported by the kernel. Change into the directory and do a directory listing. Try edit or delete files on your `/demo` mount point..



- What happens if you try run these commands as `root` either directly or via `sudo`?
- What scenarios could this be useful to have as a mount option?

You can of course combine these options (and many more) to achieve a number of outcomes. These include the use of options to optimise file system access speeds, or minimise the number of writes occurring to a disk, as well as controlling synchronous vs. asynchronous writes. The `nodelv` option is particularly interesting as it can circumvent approached by an attacker to create raw *device nodes* outside of the `/dev` file system.

You are encouraged to do some reading of the common options and experiment with these to help fill out your understanding.



What combination of options would you suggest for the following scenarios:

- The `/tmp` directory on a busy web server
- A collection of static data that is being served to clients
- A file system exported from another system

Making it permanent

You of course do not need to enter these command every time you start. Once you have arrived at the correct combination of options you can make these persistent by editing the `/etc/fstab`. For details on the format you can issue the command `man 5 fstab`².



You can see what manual pages are available for a topic by running `man -k topic`. An example is shown below.

```
$ man -k fstab
fstab (5)          - static information about the filesystems
fstab-decode (8)   - run a command with fstab-encoded arguments
systemd-fstab-generator (8) - Unit generator for /etc/fstab
```

Live Changes

Sometimes you need to change the mount option flags for a particular file system without unmounting and then remounting it (for example a system is running). You can use the approach below to do this. Its important to note that if an attacker has already gained `root` privileges on your system you have much larger issues to worry about. As such, while filesystem based protections can provide a significant defence against remote attackers or malware, they can be bypassed if the system is compromised.

```
mount -o remount,bind,ro olddir newdir
```

Note that a read-only bind will create a read-only mountpoint (VFS entry), but the original filesystem superblock will still be writable, meaning that the `olddir` will be writable, but the `newdir` will be read-only.

It's also possible to change `nosuid`, `nodev`, `noexec`, `noatime`, `nodiratime` and relative VFS entry flags by "remount,bind" operation. The another (for example filesystem specific flags) are silently ignored. It's impossible to change mount options recursively (for example with `-o rbind,ro`).

FS related sysctl

Two options that can be used for further protecting your platform that relate to the operating system are:

- `fs.protected_hardlinks`
- `fs.protected_symlinks`

Setting these to 1 provides protection against creating or following links under certain conditions. See the system documentation for more details.



Take a Backup

Make sure you have saved all your configuration and taken a snapshot of the system you can roll back to if (when?) you break things in the future.

²The specification of 5 here says you specifically want section 5 of the system manual, which deals with file formats and configuration files. `man man` for more details on this.

3 Filesystem Attributes



This task is to be done on the Ubuntu Server set up previously

Most common Linux file-systems (and specifically the ext family) support an additional set of permissions beyond the standard Read, Write and Execute (RWX) that you are familiar with. These are administered using the `chattr` command to manipulate file attributes (as opposed to permissions managed by `chmod`). A rich set of additional properties are available, as shown in the [manual](#) page for the command. The most relevant part if this is highlighted below:

The letters 'aAcCdDeFijmPsStTux' select the new attributes for the files: append only (a), no atime updates (A), compressed (c), no copy on write (C), no dump (d), synchronous directory updates (D), extent format (e), case-insensitive directory lookups (F), immutable (i), data journaling (j), don't compress (m), project hierarchy (P), secure deletion (s), synchronous updates (S), no tail-merging (t), top of directory hierarchy (T), undeletable (u), and direct access for files (x).

The following attributes are read-only, and may be listed by `lsattr(1)` but not modified by `chattr`: encrypted (E), indexed directory (I), inline data (N), and verity (V).

Note however, that **not** all flags are supported across all file-systems. A detailed breakdown on the specifications around this are summarised on [Wikipedia](#).

While this array of options is confusing at first review, these offer a wide variety of specific functionality that is useful in larger operations. For the purposes of **this course**, the most important ones from a direct security point of view (and widely supported across different filesystems) are:

- **append only (a)**
A file with the a attribute set can only be open in append mode for writing. File can only grow in size.
- **immutable (i)**
File cannot be deleted or renamed, no link can be created to this file and no data can be written to the file. Even the superuser/root is prevented, from erasing or changing the contents of the file.
- **secure deletion (s)**
In principle file is overwritten with zeros prior to deletion. Not honoured on `ext2` and `ext3` file systems. Not guaranteed on NAS/SAN and SSD media either.

Application

Manipulating these attributes revolves around attaching a set (+) or unset (-) modifier to the attribute. For example setting an IMMUTABLE flag on `/etc/ssh/sshd_config` would be:

```
$ sudo chattr +i /etc/ssh/sshd_config
```

A complementary tool to `chattr` is `lsattr` which you can use to list the set attributes for a file, directory, or group of files. From the example above, you can see the result. note that the `lsattr` does not require privileged access.

```
$ ls -la /etc/ssh/sshd_config
-rw-r--r-- 1 root root 3281 Aug 25 12:54 /etc/ssh/sshd_config
$ lsattr /etc/ssh/sshd_config
----i-----e----- /etc/ssh/sshd_config
```


Over to you.

Use the `chattr` and `lsattr` tools to undertake the tasks below, and then answer the questions. To do this you will need to set and unset attributes.

Try the following activities on some test files in your home directory.

Immutable

1. Create a file
2. Mark it immutable
3. Try delete it as the file owner (normal user).
4. Try delete it as root (privileged user).
5. Remove the immutable flag
6. Try again.



Good idea?

Many hardening guides suggest making most of the `/etc` directory IMMUTABLE as it makes them secure from accidental removal or tampering. Consider the following:

- What do you think the impact of this could be?
- What do you think any negative outcomes of this could be?
- Is this really providing security?
- What if an attacker gains root or user privileges they could remove set attributes?

Append only

1. Create a small text file containing a few lines of text
2. Mark the file as append only.
3. Open the file in an editor and remove some text.
4. Are you able to save it?
5. Try adding more text.
6. Does it work now?
7. Remove the append only flag.
8. Is the file editable as you would normally expect ?



Good idea?

Similar to immutable, many hardening guides suggest making most of the `/var/log` directory APPEND ONLY as it makes them secure from accidental removal or tampering. Consider the following:

- What do you think the impact of this could be?
- What do you think about the impact of setting logs to append only is?
- What impact might this have when logs need to be rotated?
- Is this really providing security?
- What if an attacker gains root or user privileges they could remove set attributes?

Secure Deletion

Secure Deletion is not necessarily supported on old ext variants, but is supported on ext4 which is commonly used. Think about the application of this functionality.

1. What kind of files would you use this on?
2. What could a benefit of using this be?
3. What could some disadvantages be?
4. What kinds of physical storage devices could this be most appropriate for, and why?

Discuss your view points with others in the class and make a note of your discussions.



Things to know..

Some operating systems such as the BSD family can go into a '[secure](#)' run-level where these attributes cannot be changed, making it necessary to reboot to change them and thus modify files.



Linux supports this too, sort of

Linux Kernel does support similar protections when dropping into **Security Level 3** and making use of the CAP_LINUX_IMMUTABLE. The latter if set means the system must be rebooted without the flag set for even the `root` user to be able to remove the immutable attribute. From one [article](#) discussing this:

Try to make sure that the configuration cannot be altered unauthorised. Makes it impossible to change immutable and append file attributes. Devices cannot be (un)mounted, swap devices cannot be reconfigured and `mknod()` call is disallowed. You are banned to set the time backwards or close to overflow (you can still make the time flow slower, though). Hostname and domain name cannot be changed, nor can be the `printk` logging level. (leftmargin=4cm)

4 PAM Auditing



This task is to be done on the Ubuntu Server set up previously



Before commencing with this task, make sure you take a snapshot.

Getting users to set a good password is a challenge. Fortunately as an administrator you can implement some basic *quality* check using the PAM authentication framework (as we used in setting the TOTP 2FA in week 2). This can be illustrated using the Linux PAM framework, but these same principles apply to a wide variety of other systems that use passwords. There are no other tasks this week or next week that have a dependency on this one.

One of the more popular approaches to doing this is to make use of `libpam_cracklib`³. This used `cracklib` as the underlying workhorse to do the validation. The `libpam_cracklib` module undertakes password quality checks in the following manner. The first check is the `Cracklib` routine is called to check if the password is part of a dictionary; if this is not the case an additional set of strength checks is done. These checks are:

1. Is the new password a palindrome?
2. Is the new password the the old one with only a change of case?
3. Is the new password too much like the old one?
This is primarily controlled by the `difok` argument which is a number of character changes (inserts, removals, or replacements) between the old and new password that are enough to accept the new password. The default value is **five (5)** changes.
4. Is the new password too small?
This is controlled by six arguments:
`minlen`, `maxclassrepeat`, `dcredit`, `ucredit`, `lcredit`, and `ocredit`.
5. Is the new password a rotated version of the old password?
6. Optional check for same consecutive characters.
7. Optional check for too long monotonic character sequence.
8. Optional check whether the password contains the user's name in some form.



`cracklib` sourcecode and sample dictionaries can be found on Github at:
<https://github.com/cracklib/cracklib>

³Details of configuration can be found in the [manual](#) page.

Setup



Before starting this

1. You should make a copy of your existing configuration for PAM
2. Disable the TOTP login requirement
3. Configure ssh to allow password authentication (ie. back to default)
4. Confirm accounts can login successfully



- Install the `libpam-cracklib` on your Ubuntu server using `apt`.
- It is strongly recommended to create a temporary user to test this all with. The examples below make use of a user called `passdemo` which was created with a very simple password 'foo'.
- Make sure you can log in as this user and change the password.

Configuration

Now that we have things set up we can start configuring the cracklib.

1. You need to edit `/etc/pam.d/common-password`
Make sure you have a backup of the original file before doing this.
2. Add the following lines to the **bottom** of the `/etc/pam.d/common-password` file

```
#CND cracklib
#password requisite pam_cracklib.so retry=3 minlen=8 difok=3
```

3. Duplicate the last line, remove the `#` and change the path after the `pam_cracklib.so` to read:

```
retry=3 minlen=16 difok=3 ucredit=-1 lcredit=-2 dcredit=-2 ocredit=-2
```

```
password requisite pam_cracklib.so retry=3 minlen=8 difok=3
```

4. Change the last part of the line (or copy and create a new line, commenting out the previous one) to read as:

```
pam_cracklib.so retry=3 minlen=16 difok=3 ucredit=-1 lcredit=-2 dcredit=-2 ocredit=-2
```

The values given are suggested, and you may need to change/adjust these values as per your setup (especially in the real world) to avoid brute force attack. The meanings of these parameters are detailed below.

`retry=3` Prompt user at most 3 times before returning with error. The default is 1.

`minlen=16` The minimum acceptable size for the new password.

`difok=3` This argument will change the default of 5 for the number of character changes in the new password that differentiate it from the old password.

`ucredit=-1` The new password must contain at least 1 uppercase characters.

`lcredit=-2` The new password must contain at least 2 lowercase characters.

`dcrit=-2` The new password must contain at least 2 digits.

`ocredit=-2` The new password must contain at least 2 symbols.



Positive vs. Negative

NEGATIVE number such as -2 (e.g. `ucredit=-2`) indicates that this is the **minimum** number of upper case letters that must set a new password. The POSITIVE number is the **maximum** credit for having digits in the new password. If you have less than or N digits, each digit will count +1 towards meeting the current `minlen` value. The default for `dcrit` is 1 which is the recommended value for `minlen` less than 10.

Testing

1. Log in as your test user.
2. Try set a new password.
3. you may find this challenging
4. Use the guidance provided above as to what the restrictions are enforcing
5. Write down what the restrictions are
6. Using the decoded restrictions: **Find a suitable password that satisfies the requirements and set this as the users password. Write this password down.**
7. You can verify your configuration using the exemplar passwords provided below.



Errors?

If you get an error similar to that below, this means you have a typo, or the module is not installed. Check your work carefully.

```
passdemo@cnd22: $ passwd
Changing password for passdemo.
Current password:
passwd: Module is unknown
passwd: password unchanged
```



Testing OK

If you need to check your configuration, you can use either of these two passwords that meet the criteria: `5I!LOve@PiaZza6YeS` or `s~8Kn<bzg9Ruv,8s`
Do you think these are suitably memorable passwords, especially the latter?



Password restrictions

Now that you have successfully managed to set a password, consider the following:

- Can you verify that the `difok` parameter works?
- What value do you think `difok` brings
- Do you think these parameters are reasonable and allow users to give suitably memorable passwords ?
- Do you think you can come up with your own set of parameters you feel are more reasonable?

Cleanup

If you need to remove `libpam-cracklib` along with configuration, data, and all of its dependencies, you can use the following command:

```
$ sudo apt-get -y autoremove --purge libpam-cracklib
```



NOTE: You will need to copy back your original configuration file after doing this. If you fail to do so you may well end up unable to log in.



`libpam-pwquality`

An alternate to `libpam-cracklib` is `libpam-pwquality`. To trial this, you must **uninstall** `libpam-cracklib`.

The `libpam-pwquality` PAM module provides common functions for password quality checking and also scoring them based on their apparent randomness. The library also provides a function for generating random passwords with good pronounce-ability. you can explore the use of this and compare with the `libpam-cracklib` functionality.

5 SSH Data

Sometimes existing system logging can reveal interesting trends. In this Task we will be looking at syslog (system log) messages generated by the the Secure Shell Daemon (SSHD) on a Linux system. SSH provides a secure way of logging into and transferring files to remote systems. While primarily this provides a command line interface, there are tools which build on top of the protocol to allow file transfer - such as [FileZilla](#).

An extract of these kind of logs is as follows:

```
Apr 7 12:03:25 trollhelm sshd[2199135]: Failed password for root from 157.245.53.136 port 43378 ssh2
Apr 7 12:03:27 trollhelm sshd[2199135]: Received disconnect from 157.245.53.136 port 43378:11: Bye Bye [preauth]
Apr 7 12:03:27 trollhelm sshd[2199135]: Disconnected from authenticating user root 157.245.53.136 port 43378 [preauth]
Apr 7 12:03:27 trollhelm sshd[2199137]: Invalid user donovan from 43.128.1.222 port 52592
Apr 7 12:03:27 trollhelm sshd[2199137]: pam_unix(sshd:auth): check pass; user unknown
Apr 7 12:03:27 trollhelm sshd[2199137]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=43.128.1.222
Apr 7 12:03:29 trollhelm sshd[2199139]: Invalid user barret from 152.67.47.195 port 42948
Apr 7 12:03:29 trollhelm sshd[2199139]: pam_unix(sshd:auth): check pass; user unknown
Apr 7 12:03:29 trollhelm sshd[2199139]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=152.67.47.195
Apr 7 12:03:29 trollhelm sshd[2199141]: Invalid user ts3bot from 49.234.43.39 port 48080
Apr 7 12:03:29 trollhelm sshd[2199141]: pam_unix(sshd:auth): check pass; user unknown
Apr 7 12:03:29 trollhelm sshd[2199141]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=49.234.43.39
Apr 7 12:03:30 trollhelm sshd[2199143]: Invalid user dita from 64.225.119.164 port 34330
Apr 7 12:03:30 trollhelm sshd[2199143]: pam_unix(sshd:auth): check pass; user unknown
Apr 7 12:03:30 trollhelm sshd[2199143]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=64.225.119.164
Apr 7 12:03:30 trollhelm sshd[2199137]: Failed password for invalid user donovan from 43.128.1.222 port 52592 ssh2
Apr 7 12:03:30 trollhelm sshd[2199139]: Failed password for invalid user barret from 152.67.47.195 port 42948 ssh2
Apr 7 12:03:31 trollhelm sshd[2199141]: Failed password for invalid user ts3bot from 49.234.43.39 port 48080 ssh2
Apr 7 12:03:31 trollhelm sshd[2199137]: Received disconnect from 43.128.1.222 port 52592:11: Bye Bye [preauth]
Apr 7 12:03:31 trollhelm sshd[2199137]: Disconnected from invalid user donovan 43.128.1.222 port 52592 [preauth]
Apr 7 12:03:31 trollhelm sshd[2199143]: Failed password for invalid user dita from 64.225.119.164 port 34330 ssh2
Apr 7 12:03:31 trollhelm sshd[2199145]: Invalid user space from 183.62.25.218 port 17045
Apr 7 12:03:31 trollhelm sshd[2199145]: pam_unix(sshd:auth): check pass; user unknown
Apr 7 12:03:31 trollhelm sshd[2199145]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=183.62.25.218
Apr 7 12:03:31 trollhelm sshd[2199139]: Received disconnect from 152.67.47.195 port 42948:11: Bye Bye [preauth]
Apr 7 12:03:31 trollhelm sshd[2199139]: Disconnected from invalid user barret 152.67.47.195 port 42948 [preauth]
Apr 7 12:03:32 trollhelm sshd[2199143]: Received disconnect from 64.225.119.164 port 34330:11: Bye Bye [preauth]
Apr 7 12:03:32 trollhelm sshd[2199143]: Disconnected from invalid user dita 64.225.119.164 port 34330 [preauth]
Apr 7 12:03:32 trollhelm sshd[2199141]: Received disconnect from 49.234.43.39 port 48080:11: Bye Bye [preauth]
Apr 7 12:03:32 trollhelm sshd[2199141]: Disconnected from invalid user ts3bot 49.234.43.39 port 48080 [preauth]
Apr 7 12:03:34 trollhelm sshd[2199145]: Failed password for invalid user space from 183.62.25.218 port 17045 ssh2
Apr 7 12:03:35 trollhelm sshd[2199145]: Received disconnect from 183.62.25.218 port 17045:11: Bye Bye [preauth]
Apr 7 12:03:35 trollhelm sshd[2199145]: Disconnected from invalid user space 183.62.25.218 port 17045 [preauth]
Apr 7 12:03:43 trollhelm sshd[2199147]: Invalid user admin from 101.227.5.66 port 51984
Apr 7 12:03:43 trollhelm sshd[2199147]: pam_unix(sshd:auth): check pass; user unknown
Apr 7 12:03:43 trollhelm sshd[2199147]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=101.227.5.66
Apr 7 12:03:46 trollhelm sshd[2199147]: Failed password for invalid user admin from 101.227.5.66 port 51984 ssh2
Apr 7 12:03:48 trollhelm sshd[2199147]: Received disconnect from 101.227.5.66 port 51984:11: Bye Bye [preauth]
Apr 7 12:03:48 trollhelm sshd[2199147]: Disconnected from invalid user admin 101.227.5.66 port 51984 [preauth]
Apr 7 12:03:51 trollhelm sshd[2199149]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=61.177.173.25 user=root
Apr 7 12:03:54 trollhelm sshd[2199149]: Failed password for root from 61.177.173.25 port 38092 ssh2
Apr 7 12:03:57 trollhelm sshd[2199151]: Invalid user oracle from 106.54.17.221 port 50046
Apr 7 12:03:57 trollhelm sshd[2199151]: pam_unix(sshd:auth): check pass; user unknown
Apr 7 12:03:57 trollhelm sshd[2199151]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=106.54.17.221
```



Files for this task can be found in **Task5.zip** within the databack for this week. This file is available on Moodle and the password is provided in Lectures. Remember to validate the checksums of the Task5.zip:

MD5 299c98ebfb9108ed37af93718efb9ecc

SHA256 43e4c4abaf7c9525d2303c3915ff905e8acef483545e41d0373b345f11738a18

You should extract and decompress the `auth.sample` in the task zip file so that you can follow along with the examples and structured analysis following. Files can either be downloaded directly to your VM, or you can copy them up using FileZilla (remember to have installed the `openssh-server` package using `apt` if you choose this method)

An initial observation of these logs show a number of errors. This is a case of where existing operational system logs can be used as a form of inadvertent source of Threat Intelligence, where logs can be processed for further information.

Unstructured Data Analysis

While the `syslog`⁴ format is fairly regular, a large part of it is unstructured, with fields beyond the first few left up to each application developer. This can be treated as regular text files with fields separated by a space, and no special tools are needed.



SSH logs

The ssh log files are produced by the SSH Daemon⁵ (server) application. These messages are received by the `syslog` daemon, which is responsible for writing the log files to disk. `syslog` handles logging from many other system processes. Some applications, such as Apache, write to their own log files directly. This works when only one application writes to a log. The SSH log messages are normally placed in a file called `auth.log` in `/var/log/`. The fields in the files we are dealing with are defined as follows:

timestamp process[pid]: <application log entry>

How big is it?

The `wc` command provides a useful way of seeing how big a file is in terms of lines when using the `-l` flag. This can be used in combination with other tools such as `du` and `ls`.

```
$ wc -l auth.sample
462809 auth.sample
```

Invalid Users?

One type of entry noticed is that there are messages containing the phrase `invalid user`. How many of these are present? We can use the `fgrep` command to search for a string in the text file (lines wrapped for readability):

```
$ fgrep "invalid user" auth.sample
Apr  4 00:00:20 trollhelm sshd[2084710]:
Failed password for invalid user jenkins from 150.95.30.250 port 40402 ssh2
Apr  4 00:00:21 trollhelm sshd[2084710]:
Disconnected from invalid user jenkins 150.95.30.250 port 40402 [preauth]
Apr  4 00:00:28 trollhelm sshd[2084712]:
Failed password for invalid user mcserver from 165.22.50.136 port 51690 ssh2
Apr  4 00:00:30 trollhelm sshd[2084712]:
Disconnected from invalid user mcserver 165.22.50.136 port 51690 [preauth]
Apr  4 00:00:31 trollhelm sshd[2084714]:
Failed password for invalid user teamspeak3 from 175.6.65.68 port 46389 ssh2
Apr  4 00:00:33 trollhelm sshd[2084716]:
Failed password for invalid user student4 from 167.71.58.244 port 36978 ssh2
....
```

We can easily count this by combining `fgrep` with `wc`.

```
$ fgrep "invalid user" auth.sample | wc -l
102567
```

This is clearly quite a large number of lines, and not feasible for manual inspection. The data could do with some cleaning up before further processing.

⁴<https://en.wikipedia.org/wiki/Syslog> - details on the format can be found [here](#) and [here](#).

⁵The term **Daemon** was originally coined at MIT in the early 1960's

Extracting information?

For the purposes of getting a list of users we can trim out quite a bit of information that is not needed. One advantage of the syslog's regular format is we can easily remove the leading timestamp and sshd indicator (after ensuring that this process is relevant to all our lines) by using `cut` to remove the columns we don't need. In this case we only return from column 42 onward:

```
$ fgrep "invalid user" auth.sample | cut -c42-
Disconnected from invalid user teamspeak 118.24.231.241 port 34258 [preauth]
Failed password for invalid user school from 139.59.35.114 port 44680 ssh2
Disconnected from invalid user school 139.59.35.114 port 44680 [preauth]
Failed password for invalid user git from 218.78.54.80 port 46217 ssh2
Disconnected from invalid user git 218.78.54.80 port 46217 [preauth]
Failed password for invalid user bct from 82.156.28.70 port 45330 ssh2
Disconnected from invalid user bct 82.156.28.70 port 45330 [preauth]
....
```



Cutting Columns

A challenge when cutting columns can occur when column values change due to the size of other fields in the file. In the case of these logs one of the areas that can change is the `sshd[pid]` where *pid* is the Process id of the particular ssh daemon. While this can be very useful for tracing transactions in a busy log file, in many cases its information not needed, and often it gets in the way of regular processing. One way of dealing with this is to replace this column with an appropriate token. This can for example be done using `sed`:

```
echo "Apr 4 00:00:31 trollhelm sshd[2084714]:" | sed -e 's/sshd\[.*\]/sshd/g'
Apr 4 00:00:31 trollhelm sshd:
```

This then provides a predictable and consistent field size. `sed` can similarly be used to delete unneeded data from a identifier to the end of line. Compare this approach with making use of fields by their position, rather than absolute column number.

We can see here that the usernames are in positions five (5) and six (6) in the different lines. For simplicity we can look at just where there is a failed password for a user. The `awk` command is a great way of extracting fields. here we extract the 6th file which is the username. The `head` command provides an easy way to just show a limited number of files.

```
$ fgrep "invalid user" auth.sample | cut -c42- | grep "^Failed password" | awk '{print $6}' | head
jenkins
mcserver
teamspeak3
student4
admin
dbuser
redmine
zxcloudsetup
admin
tester
```



How would you modify the process above to extract the username from the **Disconnect from** lines ?

Removing duplicates

Running the command from the previous step still gives us 51 159 lines of output which is a pain to deal with, and we can reduce the data further by in this case looking for lines starting with (as indicated by the ^ symbol) "Failed password".

```
$ fgrep "invalid user" auth.sample | cut -c42- | grep "^Failed password" | awk '{print $6}' | wc -l
51159
```

Fortunately there are a number of tools which can make our life easier when processing logs. The `uniq` command in conjunction with `sort` provide sorting and filtering of data.

```
$ fgrep "invalid user" auth.sample | cut -c42- | grep "^Failed password" | awk '{print $6}' > invalid.tmp
# this line writes out long list of users to a file called invalid.tmp
$ cat invalid.tmp | sort | uniq > invalid.uniq
$ wc -l invalid.uniq
7692 invalid.uniq
$ head invalid.uniq
$(ping
,
*****
0
000
0000
01
0101
017
02
```

This starts providing us with more interesting information and we have reduced our list to 7 692 entries.



Debugging? Sometimes things do not go as planned, a good piece of advice is to remove items from the pipeline. It is similarly **highly recommended** to validate that what you are getting out of a pipeline is what you are expecting. This is **especially important** when using functions such as `uniq` which aggregates data.

Active use?

Now we have a list of usernames it would be useful to know which are the most commonly tried. This requires very little modification to what we did in the previous step. We make use of the `-c` (count) flag to `uniq` and then run the output through another `sort` this time providing the parameters of `-rn` (sort numerically in reverse order) to `sort`.

```
$ cat invalid.tmp | sort | uniq -c | sort -rn > invalid.uniq.cnt
$ head invalid.uniq.cnt
1897 admin
1021 test
702 user
562 ubuntu
481 postgres
450 oracle
443 ftpuser
428 git
258 guest
235 nagios
```



How could you take the above approach and extract the top IP addresses ?

**Need help?**

If you want to get help on any Linux command, you can use the command `man cmd` to access its manual page. You can even try `man man`. The manual pages describe the various command line options command support.

**Pattern matching bites..**

When using pattern matching with tools such as `grep`, one needs to be aware of their strengths and weaknesses. A common source of errors is where lines are extracted because a phrase appears as part of another. If you are searching for phrases bounded by whitespace, you should consider using the `-w` flag to `grep` to indicate so.

6 Deliverable

In addition to completing the preceding tasks, you should make sure that you have saved a document detailing your approach to answering the tasks below. This is for **your use**, and you are expected to have it available when you write the **Engagement Quiz 2**. It is strongly suggested that you keep text (i.e. that you can copy and paste) versions of your output and commands, in addition to screenshots if you use them. You are encouraged to discuss your approaches and findings with others in your class.

Analysis Task

You have been provided with an extract of log files from an operational system in the file `sshd_log.txt`. These entries may have a slightly different format to those in the example above, but the principle is the same. This file can be found in the zip for the preceding task.

You should base your analysis on what was covered in the previous task, as well as using the skills around text processing developed in Tutorial 2. The data here as noted previously is not as neatly structured as the regimented csv or other regular data files, and is more representative of the kind of logging data that one has to deal with in real-world systems.



The short log extract for this task can be found in **deliverable-week3.zip** within the datapack for this week. This file is available on Moodle and the password is provided in Lectures. Remember to validate the checksums for the file:
MD5 d3a9c13b53640825da4aea066eae3335
SHA256 ef196cd20451c1457742ed642456d80491b2767931f7a164ca25efe58279359d



HINT: While this file is small enough for processing in Excel you may find tooling on the command line a lot easier, especially as data volumes rise. Avoiding Excel is strongly encouraged.



Comparisons

The `-f` and `-o` flags for `grep` can be of use when doing comparisons (especially of lists) and reduce the amount of work needed. Take care that the files may need to be formatted appropriately. This can sometimes cause issues if comparing very large files in which case alternate approaches may be needed. See man pages for details.



You need to conduct some analysis on this file and answer the questions below. Keep track of how you did this as this will prove useful in the future. File is in standard `syslog` format.

- How many lines does the file contain?
- How many different **types** of message are present?
HINT: This is not straight forward and may require a number of iterations of filtering. The purpose of this is to start working out what message parts are static and where information changes. Discard information that changes.
- How many hosts connected to this system?
HINT: Consider the records present for the host 200.54.189.102. What do these tell us ?
- One group of hosts stands out as there being a large number from within the same /16, and specifically a single /24 netblock. What are the /16 and the /24 networks?
- Which organisation owns these blocks?

- (f) Which Autonomous System Number (ASN)⁶ does this netblock belong to?
- (g) What do you think the reason for this could be?
- (h) What were the top 10 hosts, based on the number of connection attempts?
- (i) Build up a list of usernames tried against this system. How many are there?
- (j) What were the top 10 usernames (by number of occurrences) attempted?
- (k) For the username `pi`, how many different (unique) hosts tried this?
- (l) How many of the usernames appear on the list of default accounts used by the Mirai IOT worm?
HINT: This means searching for each of the names that exist in your list against those appearing in the Mirai list. There are two different approaches that can be used with this.
- (m) Building on your solutions for h) and j) produce a script called `details.sh`. This should be able to be executed with a filename (you can assume that it will have the same format as you are working with) passed on the command line, and it should produce a listing of top 10 usernames and top 10 hosts (in both cases based on the number of occurrences in the file). For example: `details.sh sshlog-june22.log`



The Mirai worm source code is on [GitHub](#). You **do not** need to understand the C language its written in but you will be able to spot the usernames tried. you will need to extract these to a suitable text file for processing. There are other sources where you can find the username list.



Automation?

What do you think the advantage is of stringing several commands together is compared to writing a dedicated program to process data? How would you save your list of commands so you could easily process data like this in the future without having to type them in one by one?



Want to do more?

Repeat the questions above with the `auth.sample` file. How many of your commands work 'as is' with the other data file?

Deliverable Items



Confirm you have the following complete in terms of specific deliverables that you will need to have completed:

1. Your notes from the SSH log processing activity above
2. Your script as the final point in the log processing

Make sure you have the above carefully saved (along with your PGP private key which you will need in due course).

As a final check, have you made snapshot of your VM (and a (offline) backup?)

⁶An [ASN](#) is a unique number allocated to an organisation and used as part of the global routing of the public internet via BGPv4. An AS can be related to multiple netblocks for both IPv4 and IPv6

A News

The following are new articles of interest in the last week. These are all relatively short, and you should get in the habit of staying aware of what is going on in the industry at large. The topics discussed here fall in scope for the engagement tasks.



News Week of 4 September 2022

- ATTACK** [Time keeps on slippin' slippin' slippin': The 2023 Active Adversary Report for Tech Leaders](#)
The midyear Active Adversary Report for Tech Leaders, analyzing data amassed by Sophos' Incident Response team and covering the first half of calendar year 2023. This is the second Active Adversary Report of the year.
- APT** [Lazarus Group's infrastructure reuse leads to discovery of new malware](#)
In the new Lazarus Group campaign we recently disclosed, the North Korean state-sponsored actor continues to use much of the same infrastructure despite those components being well-documented by security researchers over the years. Their continued use of the same tactics, techniques and procedures (TTPs) — many of which are publicly known — highlights the group's confidence in their operations and presents opportunities for security researchers.
- ATTACK** [APT Group use Legit Software in Supply Chain Attack](#)
A previously unknown advanced persistent threat (APT) group used the legitimate software to carry out a supply chain attack with the goal of deploying a backdoor onto victim computers.
- VULNS** [Traders' Dollars in Danger: CVE-2023-38831 zero-Day vulnerability in WinRAR exploited by cybercriminals to target traders](#)
While researching the spread of malware the unit came across a previously unknown vulnerability in the processing of a popular archive format by WinRAR. By exploiting a vulnerability within this program, threat actors were able to craft ZIP archives that serve as carriers for various malware families.
- CRIME** [Montenegro and its allies are working to recover from the massive cyber attack](#)
A teenage member of the Lapsus\$ hacking group was on Wednesday found guilty by a London jury to have hacked Uber (UBER.N) and fintech firm Revolut.

The above articles are **in scope** for Engagement Activity 1

B Listening

The links below are provided as a value add only and are not a component of future engagement quizzes.



A major discussion in the computing and security spheres that has gained traction in the last few years is that of the *Right to Repair*. A seminal talk by Sick Codes, on the matter was given at last Year's DEFCON conference. The talk has now been released on Youtube. The second video is some interesting background as to why one vendor of equipment in particular has been a target of exploration and exploitation.

- [DEF CON 30 - Sick Codes - Hacking the Farm = Breaking Badly into Agricultural Devices](#) ⌚ 47:58
- [Farmers Are Hacking Their Tractors Because of a Repair Ban](#) ⌚ 11:29



This talk, by a group of high-school students, on manipulating cards used for public transport made quite a splash at this year's DEFCON conference last month. The video has just been released.

- [DEF CON 31 - Infinite Money Glitch - Hacking Transit Cards](#) ⌚ 45:04



The Billion Dollar Heist

The story of one of the most daring cyber heists of all time, the Bangladeshi Central Bank theft. Tracing the origins of cyber-crime, from basic turn of the millennium credit card fraud by individuals to global criminal organizations. If you have existing subscriptions to the services below it may be included, alternatively you are encouraged to get a few friends together and rent it.

- [Billion Dollar Heist — Official Trailer \(2023\)](#) ⌚ 2:06
- [Watch on TV2.no](#) (subscription required) Rental:NOK 49
- [Watch on Apple TV](#) (subscription required) Rental:NOK 49
- [Watch on ViaPlay](#) (subscription required) Rental:NOK 49

C FS Notes

Linux Filesystem Creation This addendum is to be used in conjunction with Task 2, and provides the basics for the setup, creation and population of a additional file system to use in the the aforementioned Task. Take care to follow all the steps carefully. You are **strongly** encouraged to read though this in **full** before starting.



Before proceeding make sure you have a snapshot of your system. Some of the changes to be made can have a potentially negative impact. Where you edit files, make sure to keep a back up of them.

Disk Creation

You need to create and attach a new disk (storage) to your Ubuntu Server we created in Week 1. This will allow us to demonstrate and explore a number of features with minimal risk to the base system.



You need to complete the following:

1. Shut down your Server VM
2. In VMware workstation, go to the VM option and then *Settings* (Ctrl+D) and click *Add*.
3. Select a hard disk and accept the defaults offered, changing the size to 0.5 GB. This will be more than sufficient in size for this exercise.
4. Start your machine
5. Check that the drive is showing up by running the command below

NOTE: The output will not be exactly as below as device numbering and values depends a lot on individual systems but what you see should be very similar.

```
$ dmesg | grep sd
[ 7.009666] sd 32:0:0:0: [sda] 41943040 512-byte logical blocks: (21.5 GB/20.0 GiB)
[ 7.010162] sd 32:0:0:0: [sda] Write Protect is off
[ 7.010520] sd 32:0:0:0: [sda] Mode Sense: 61 00 00 00
[ 7.010725] sd 32:0:0:0: [sda] Cache data unavailable
[ 7.010751] sd 32:0:0:0: Attached scsi generic sg1 type 0
[ 7.011088] sd 32:0:0:0: [sda] Assuming drive cache: write through
[ 7.012646] sd 32:0:1:0: [sdb] 1048576 512-byte logical blocks: (537 MB/512 MiB)
[ 7.012755] sd 32:0:1:0: Attached scsi generic sg2 type 0
[ 7.013299] sd 32:0:1:0: [sdb] Write Protect is off
[ 7.013669] sd 32:0:1:0: [sdb] Mode Sense: 61 00 00 00
[ 7.014027] sd 32:0:1:0: [sdb] Cache data unavailable
[ 7.014345] sd 32:0:1:0: [sdb] Assuming drive cache: write through
[ 7.022022] sd 32:0:1:0: [sdb] Attached SCSI disk
[ 7.022945] sda: sda1 sda2 sda3
[ 7.024296] sd 32:0:0:0: [sda] Attached SCSI disk
```

I you get an error accessing dmesg have you completed Task 1?

You an see the second drive showing up and listed as:

```
[sdb] 1048576 512-byte logical blocks: (537 MB/512 MiB
```

in the output above.

You can also list the block (storage) devices that the Linux kernel can see:

```
$ lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0                              7:0      0   62M  1 loop /snap/core20/1611
loop1                              7:1      0  79.9M  1 loop /snap/lxd/22923
loop2                              7:2      0  103M  1 loop /snap/lxd/23541
loop3                              7:3      0  63.2M  1 loop /snap/core20/1623
loop4                              7:4      0   47M  1 loop /snap/snapd/16292
loop5                              7:5      0   48M  1 loop /snap/snapd/16778
sda                                8:0      0   20G  0 disk
├─sda1                             8:1      0    1M  0 part
├─sda2                             8:2      0  1.8G  0 part /boot
└─sda3                             8:3      0 18.2G  0 part
   └─ubuntu--vg-ubuntu--lv 253:0    0   10G  0 lvm  /
sdb                                8:16     0  512M  0 disk
sr0                                11:0     1   1.4G  0 rom
```



In both cases you should see your drive showing up as `sdb` indicating it is the second drive. `lsblk` has the advantage of in the default configuration it does not require privileged access. **If your drive comes up as something different you should replace the name appropriately in the examples following.**

[Some explanation on storage device naming](#) is available if you would like to know more.

Partitioning

You now need to prepare it for use. We will use the simpler and older MBR partitioning format - thus the use of `fdisk` (rather than `parted` which one can use for creating more modern GPT/UEFI partition tables).



It is important you get the following commands correct and ensure you are operating on the correct (second) drive. If you apply these elsewhere you risk erasing your primary disk.

Have you made a snapshot?

```
$ sudo fdisk /dev/sdb
```

```
Welcome to fdisk (util-linux 2.37.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.
```

```
Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x15edc233.
```

```
Command (m for help): p
Disk /dev/sdb: 512 MiB, 536870912 bytes, 1048576 sectors
Disk model: VMware Virtual S
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x15edc233
```

Check the text above to make sure you are using `/dev/sdb` and that the size matches.

The following steps will then create two partitions, the first of 100MB and the second of the remainder of the disk.

```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-1048575, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-1048575, default 1048575): +100M
```

Created a new partition 1 of type 'Linux' and of size 100 MiB.

Well done, now see if the partition is there?

```
Command (m for help): p
Disk /dev/sdb: 512 MiB, 536870912 bytes, 1048576 sectors
Disk model: VMware Virtual S
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x15edc233
```

Device	Boot	Start	End	Sectors	Size	Id	Type
/dev/sdb1		2048	206847	204800	100M	83	Linux

Now repeat the process and add a second partition (partition 2) and accept all the defaults offered. After doing this you should check that both partitions (effectively logical areas of the disk) are showing.

```
Command (m for help): p
Disk /dev/sdb: 512 MiB, 536870912 bytes, 1048576 sectors
Disk model: VMware Virtual S
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x15edc233
```

Device	Boot	Start	End	Sectors	Size	Id	Type
/dev/sdb1		2048	206847	204800	100M	83	Linux
/dev/sdb2		206848	1048575	841728	411M	83	Linux

We now need to make sure the changes are written to disk, as currently they are in kernel memory only.

```
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

Take note of the **w** command which tells the **fdisk** tool to write the newly created partition tables to disk (from memory) and notifies the kernel of the change.

This completes adding the partitions to the disk. Before we can use them we need to tell the kernel they exist. We can verify that the kernel knows about them running the following command. If this does **not** show up, reboot, and recheck.

```
$ ls -lad /dev/sdb*
brw-rw---- 1 root disk 8, 16 Sep 15 13:41 /dev/sdb
brw-rw---- 1 root disk 8, 17 Sep 15 13:41 /dev/sdb1
brw-rw---- 1 root disk 8, 18 Sep 15 13:41 /dev/sdb2
```



At this point when everything is correct, take a snapshot.

Filesystem Creation

Now that we have file-systems we can format them with an appropriate file system (ext4 in this case). An example of doing this for /dev/sdb1 is shown below:

```
$sudo mkfs.ext4 /dev/sdb1
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 25600 4k blocks and 25600 inodes

Allocating group tables: done
Writing inode tables: done
Creating journal (1024 blocks): done
Writing superblocks and filesystem accounting information: done
```

Mounting

Now that we have a new formatted filesystem, we need to add it to the system so it can be accessed. Now create a mount point for your new file system (/demo) and mount it. Then verify its mounted.

```
$ sudo mkdir /demo
$ sudo mount /dev/sdb1 /demo
$ mount | grep demo
/dev/sdb1 on /demo type ext4 (rw,relatime)
```

Test Data

You should be able to change your directory to /demo and be able to create files on it. Make sure you have the following files on it as a minimum. You will need to use `sudo` (preferable) or change the permissions on the mount point to be writable by your non privileged user.):

1. A text file
2. An executable file. To aid in the experiments later you need to copy /usr/bin/id to /demo/demo1. Do the same to create /demo/demo2 and /demo/demo3 ensuring the 'x' (executable) permissions are set.
3. A simple shell script that prints out the messageHello World. Do not forget to run `chmod u+x` on the file.

/demo/demo2 and /demo/demo3 now need to **have their permissions modified** to have SUID and GID bits set. You can verify that all three demo files are the same.

```
# chmod +s demo2
# chmod g+s demo3
```

Your directory should look similar to the following:

```
ls -la
total 152
drwxr-xr-x  3 root root  4096 Sep 15 14:00 .
drwxr-xr-x 20 root root  4096 Sep 15 13:53 ..
-rwxr-xr-x  1 root root 39424 Sep 15 13:56 demo1
-rwsr-sr-x  1 root root 39424 Sep 15 13:56 demo2
-rwxr-sr-x  1 root root 39424 Sep 15 13:58 demo3
drwx-----  2 root root 16384 Sep 15 13:51 lost+found
-rw-r--r--  1 root root    4 Sep 15 13:59 notes.txt
-rwxr-xr-x  1 root root   31 Sep 15 14:00 shell.sh
```

The shell script should perform similar to the following:

```
$ ./shell.sh
Hello World
```

Run `id` as your standard user, and then run each of the three demo files noting the output which should be similar to that below. This will form the baseline for the further work we do in terms of actually securing the file system. take note of preceding the calls do the **demo** files with `./` which explicitly tells the shell to execute from the current directory rather than trying to look in the system PATH.

```
#These commands are run as a non privileged user cnd
cnd@cnd22:/demo$ id
uid=1000(cnd) gid=1000(cnd) groups=1000(cnd),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lxd)
cnd@cnd22:/demo$ ./demo1
uid=1000(cnd) gid=1000(cnd) groups=1000(cnd),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lxd)
cnd@cnd22:/demo$ ./demo2
uid=1000(cnd) gid=1000(cnd) groups=1000(cnd),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lxd)
cnd@cnd22:/demo$ ./demo3
uid=1000(cnd) gid=1000(cnd) egid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),
46(plugdev),110(lxd),1000(cnd)
```

You should verify that all three demo files are the same, as we will see differign behaviour later on.



Record your baseline

- What do you notice?
- Write down the differences you see among the output from the three demo files.
- How does their output compare with that issued by `id`?

You will use this information as you return to the filesystem security task.

Unmount your drive using `sudo umount /demo`. You cannot be in the `/demo` directory when issuing this command so issue a `cd /` or `cd ~` if you get an error then try again.



At this point when everything is correct, take a snapshot.

This ends the preparation addendum. Return to Task 2 and follow the instructions in that task.

SANS
INSTITUTE

Hex File Headers and
Regex for Forensics
Cheat Sheet v1.0
SANS Forensics
<http://computer-forensics.sans.org>
<http://blogs.sans.org/computer-forensics>
By Guy Bruneau, gbruneau@sans.org

Purpose

Forensic Analysts are on the front lines of computer investigations. This guide aims to support Forensic Analysts in their quest to uncover the truth.

How To Use This Sheet

When performing an investigation it is helpful to be reminded of the powerful options available to the investigator. This document is aimed to be a reference to the tools that could be used.

This sheet is split into these sections:

- Hex File Headers
- grep/egrep
- sort
- awk
- sed
- uniq
- date
- Windows findstr

The key to successful forensics is minimizing your data loss, accurate reporting, and a thorough investigation.

grep/egrep

grep's strength is extracting information from text files. grep operates on one or multiple files when provided with a command line argument(s) that can also include wildcards:

Example: `grep "John" addressbook`
Would return the lines that contained the "John" string in the addressbook text file

Some useful flags:

- A Print number of lines after the match
- B Print number of lines before match
- c Report number of occurrences
- f Reads one or more patterns from a file. Pattern are terminated by a newline
- h Suppress the file names on the output
- i Ignore case
- l Report matching files, not matching lines
- P Interpret pattern as a Perl Regex
- v Reverse operation: return the lines **not** matching the string

The egrep (extended grep) utility can be useful to match several possible strings at the same time (in an OR mode):

egrep "John|Peter" addressbook
grep "John\|Peter" addressbook

sort

sort, as its name implies, will sort the output. There are a few interesting options you can use:

- d Uses dictionary order. Only letters, digits and blanks.
- n will sort the output assuming it is numerical (instead of string)
- u will remove redundant line, 'uniquing' the results

Hex File Header and ASCII Equivalent			
File headers are used to identify a file by examining the first 4 or 5 bytes of its hexadecimal content.			
Filetype	Start	Start ASCII Translation	
ani	52 49 46 46	RIFF	
au	2E 73 6E 64	snd	
bmp	42 4D F8 A9	BM	
bmp	42 4D 62 25	BMP%	
bmp	42 4D 76 03	BMv	
cab	4D 53 43 46	MSCF	
dll	4D 5A 90 00	MZ	
Excel	D0 CF 11 E0	MZP (immo)	
exe	4D 5A 50 00	MZ	
exe	4D 5A 90 00	MZ	
flv	46 4C 56 01	FLV	
gif	47 49 46 38 39 61	GIF89a	
gif	47 49 46 38 37 61	GIF87a	
gz	1F 8B 08 08		
ico	00 00 01 00		
jpeg	FF D8 FF E1		
jpeg	FF D8 FF E0		
jpeg	FF D8 FF FE		
Linux bin	7F 45 4C 46	ELF	
png	89 50 4E 47	PNG	
msi	D0 CF 11 E0		
mp3	49 44 33 2E	ID3	
mp3	49 44 33 03	ID3	
OFT	4F 46 54 32	OFT2	
PPT	D0 CF 11 E0		
PDF	25 50 44 46	%PDF	
rar	52 61 72 21	Rar!	
sfw	43 57 53 06/08	cws	
tar	1F 8B 08 00		
tgz	1F 9D 90 70		
Word	D0 CF 11 E0		
wmv	30 26 B2 75		
zip	50 4B 03 04	PK	

awk

awk is an extremely useful tool, especially for parsing data structured in columns. It is straightforward to use for simple purposes. Its basic use is to select some particular columns from the output: column 1 is referred to as \$1, column 2 as \$2, etc.

The space is the default awk separator. However if you want to be able to parse data separated by some other character, e.g. ":", you can use the -F flag.

Example: echo "hello:goodbye" | awk -F: '{print \$2}'

Would return "goodbye" as an output

sed

sed is an excellent command for character substitution. Example: if you want to substitute the first occurrence of the 'a' character by an 'e':

```
echo "hallo" | sed 's/a/e/'
```

The output would be: hello

You can use the g modifier to substitute all instances:

```
echo "Hallo Jenny" | sed 's/a/e/g'
```

The output would be: Hello Jenny

uniq

The uniq command reads the input and compares adjacent lines. If two or more adjacent lines are identical, all but one is removed.

Here is a list of the most common options used with uniq:

- c Prefix line with number of occurrence
- f Avoid comparing the first N fields
- i Ignore case
- s Avoid comparing the first N characters
- u Only print unique lines

Consider this input file:

```
a
b
c
b
a
c
1 a
2 b
1 c
```

Now run uniq on it: sort testfile | uniq

Now run uniq -c on it:

Date

Check the date man page for more options.

Returns the real date from epoch time:

```
date -d @1284127201
```

Return an epoch time of 1288756800:

```
date +%s -d "2010-11-03"
```

Return a 2 days old date:

```
date --date="-2 days" +%Y-%m-%d"
```

Return 20:00 hours:

```
date -d @1288310401 +%k:%M
```

Windows findstr

The Windows findstr has one interesting feature that differs from grep. If you need to search for multiple strings, you need to separate them with a space.

For example, you want or need to look for a match for WHITE or GREEN in a text file, you write your command like this:

```
findstr "WHITE GREEN" textfile
```

To make the search case insensitive, add the /I to print all variant of WHITE or GREEN.

Windows findstr Command List

- /B Matches pattern if at the beginning of a line.
- /E Matches pattern if at the end of a line.
- /L Uses search strings literally.
- /R Uses search strings as regular expressions.
- /S Searches for matching files in the current directory and all subdirectories.
- /I Specifies that the search is not to be case-sensitive.
- /X Prints lines that match exactly.
- /V Prints only lines that do not contain a match.
- /N Prints the line number before each line that matches.
- /M Prints only the filename if a file contains a match.
- /O Prints character offset before each matching line.
- /P Skip files with non-printable characters.

App Hardening

Computer Network Defence - 2023/24

Version 1.02 25/09/2023

Following on from ACCESS CONTROL, and the Initial Operating system level hardening and system configuration you undertook last week, we explore starting to focus on the security of computing systems. For our exploration of Application hardening we will be using is a Web server running the LAMP stack, and an associated web application being Wordpress.

While the page count this week is longer than previous weeks, the volume of 'work' is substantially reduced. Layout has been spaced to improve readability and content enriched to provide more examples, background and guidance relating to the concepts at hand. A number of pages provide example out put rather than problems to be solved. Use this week as an opportunity to ensure you are up to date.

Linking back to the lecture, a number of mechanisms are considered as to how to build secure foundations for application deployment. The tutorial is largely constructed as a series of steps to be followed, with a reduced content relating to 'problem solving'. Your attention is directed to the various questions for you to consider as you work though the material.

The practical tasks this week consist of three distinct groupings:

CERTIFICATES *[Tasks1-2]* Digital certificates, particularly those in the x.509 format have become a cornerstone of modern communications and e-commerce in particular. The use of a digital certificate is expected for any kind of commerce transaction that is undertaken today. In addition by using these, we are able to over TLS commonly referred to as SSL, to provide secure communication paths between systems that can be dynamically negotiated. This common naming is however incorrect as the SSL protocol was a pre-cursor , that while it offered similar functionality it is defunct. The certificates are often referred to as SSL certificates as

well. **This setup is relied on in coming weeks.**

WEBSERVER SECURITY [Tasks 3-4] As discussed in lectures, building defences is best achieved as a multi-layer approach. These tasks focus on some of the security that can be offered by the web-server. While the focus is specifically on Apache, these principles map to other web-servers.

APPLICATION [Tasks 5 & 6] The final components (including the deliverable) build on the application of knowledge and in particular the skills you gained in the first week. Regular expressions are a core skill you need to be able to work with any kind of text data. In the cybersecurity space this generally means Log data, but will typically also include emails, webpages or even network packets at times. We have already done some text processing and you should see the difference between processing structured data (as in week 1) and unstructured as in the ssh logs as done last week. Regular expressions can make the latter much easier and faster to process. However with great power come some complexity. This is **not** a defence specific task but applies across **all** subjects and disciplines. The concepts do not however seem to have been covered in any depth so far in your degree, and are being revisited here.

READING There are five brief news articles to review, along with a reading and listening activity.

These areas can all be worked on independently, with no inter-dependency between the sections. The order in which you approach these four areas is up to you. You are strongly encouraged to switch between tasks if you get stuck or find your concentration drifting. The Deliverable summarises specific questions and tasks you are expect to have worked though and solved as part of the week. The activities therein build on the concepts covered and skills developed in the preceding tasks, as well as your prior weeks. Deliverable.

We will continue to build on the configured server system, exploring on aspects of Operating System, Application and Network security. You are **strongly encouraged** to make sure you are up to date and aim to have all the content reviewed and tasks in weeks 1-3 completed before the next lecture.



This is an opportune time to refresh your networking skills as we start diving into much more network related concepts and activities from next week.

This tut *may* be updated during the course of the week with additional support material, make sure you have the latest version as shown on Moodle.

Contents

1	Setting up your own CA	1
2	Apache TLS	8
3	HTTP Headers	12
4	HTTP Security Headers	15
5	Regular Expressions	20
6	Deliverable	24
	Resources	25
A	News	26
B	Reading	27
C	Grep Features	28

CHANGELOG

- ➦ v1.0 - Initial Release
 - ➦ v1.01 - Fixed missing News/Reading Section
 - ➦ v1.02 - Added further clarification of outputs from Deliverable, reviewed Task 2. Updated News, typo in url and additional new story update.
-

1 Setting up your own CA

While TLS certificates (often referred to incorrectly as SSL certificates) are available from a variety of commercial vendors, or even for Free via the [LetsEncrypt](#) these are not viable for demo purposes, or for use on private IP addresses such as used in the tutorial. The letsencrypt/ACME certificates require direct internet connectivity.

To solve this we need to set up our own CERTIFICATE AUTHORITY (CA) to sign and thus issue appropriately certified X.509 certificates we can use of a number of purpose.



There are a number of ways of setting up the required environment one option is `tinyCA`, but this has a fairly large number of dependencies. For this task we will use the `easy-rsa` package.

The first task in this tutorial is to install the `easy-rsa` set of scripts on your CA Server. `easy-rsa` is a Certificate Authority management tool that you will use to generate a private key, and public root certificate, which you will then use to sign requests from clients and servers that will rely on your CA.

This can be installed on your Ubuntu server as below.

```
cnd@cndtut:~$ sudo apt install easy-rsa
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libccid libpcsc-lite1 opensc opensc-pkcs11 pcscd
The following NEW packages will be installed:
  easy-rsa libccid libpcsc-lite1 opensc opensc-pkcs11 pcscd
0 upgraded, 6 newly installed, 0 to remove and 0 not upgraded.
Need to get 1,365 kB of archives.
After this operation, 5,070 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```



Details and Documentation

`easy-rsa` has a very comprehensive set of documentation. The most useful to you right now is the details of [how to use it as a CA](#).

CA Initialisation

At this point you have everything you need set up and ready to use `easy-rsa`. In the next step you will create a Public Key Infrastructure, and then start building your Certificate Authority which you can issue signed certificates.



In an operational context, a CA, typically is operated on a dedicated machine with no (or at least highly restricted) network connectivity such as to prevent any compromise of the system. Certificates are moved on and off the system using removable media, and batch processing. **DO NOT** replicate this setup on an internet facing production system.

Now that you have installed `easy-rsa`, it is time to create a skeleton Public Key Infrastructure (PKI) on the CA Server. Ensure that you are still logged in as your non-root user and create an `easy-rsa` directory. Make sure that you do **not** use `sudo` to run any of the following commands, since your normal user should manage and interact with the CA without elevated privileges.

```
$ mkdir ~/easy-rsa
```

This will create a new directory called `easy-rsa` in your home folder. We'll use this directory to create symbolic links pointing to the `easy-rsa` package files that you have installed previously. These files are located in the `/usr/share/easy-rsa` folder on the CA Server. Create the symlinks with the `ln` command:

```
$ ln -s /usr/share/easy-rsa/* ~/easy-rsa/
```



Note: While other online guides might instruct you to copy the `easy-rsa` package files into your PKI directory, this tutorial adopts a symlink approach. As a result, any updates to the `easy-rsa` package will be automatically reflected in your PKI's scripts.

To restrict access to your new PKI directory, ensure that only the owner can access it using the `chmod` command:

```
$ chmod 700 ~/easy-rsa
```

Finally, initialize the PKI inside the `easy-rsa` directory:

```
$ cd ~/easy-rsa
$ ./easysrsa init-pki
```



You should see the following output if successful - replacing `cnd` with your username!
 init-pki complete; you may now create a CA or requests.
 Your newly created PKI dir is: `/home/cnd/easy-rsa/pki`

After completing this section you have a directory that contains all the files that are needed to create a Certificate Authority. In the next section you will create the private key and public certificate for your CA.

CA Building

After initialisation, you are now almost ready to actually create the public and private keys for your CA. Create a file called `~/easy-rsa/vars` and place the following lines in it, updating as appropriate:

```
set_var EASYRSA_REQ_COUNTRY    "NO"
set_var EASYRSA_REQ_PROVINCE   "Agder"
set_var EASYRSA_REQ_CITY       "Kristiansand"
set_var EASYRSA_REQ_ORG        "UC3CND101"
set_var EASYRSA_REQ_EMAIL      "barry.irwin@stud.noroff.no"
set_var EASYRSA_REQ_OU         "Community"
set_var EASYRSA_ALGO           "ec"
set_var EASYRSA_DIGEST         "sha512"
```

This will save you entering this information every time you generate a certificate going forward. When complete issue the command from within your `~/easy-rsa/` directory.

```
$ ./easysrsa build-ca
```

This will then prompt you for a passphrase to protect the CA private key and will produce output similar to the following.

Note: using Easy-RSA configuration from: `./vars`

Using SSL: openssl OpenSSL 1.1.1f 31 Mar 2020

Enter New CA Key Passphrase:

Re-Enter New CA Key Passphrase:

read EC key

writing EC key

You are about to be asked to enter information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

Common Name (eg: your user, host, or server name) [Easy-RSA CA]:CNDDemo

CA creation complete and you may now import and sign cert requests.
Your new CA certificate file for publishing is at:
/home/cnd/easy-rsa/pki/ca.crt



Do **not** worry if you get an error similar the message below. This is just due to the random seed not having been initialised. For the purposes of the tutorial this can be safely ignored. In production or for Internet facing systems this would be of some concern¹.

```
Can't load /home/cnd/easy-rsa/pki/.rnd into RNG
140409467954496:error:2406F079:random number generator:RAND_load_file:
Cannot open file:../crypto/rand/randfile.c}:98:
Filename=/home/cnd/easy-rsa/pki/.rnd
```

You now have two important files — `~/easy-rsa/pki/ca.crt` and `~/easy-rsa/pki/private/ca.key` — which make up the public and private components of a Certificate Authority.

ca.crt is the CA's public certificate file. Users, servers, and clients will use this certificate to verify that they are part of the same web of trust. Every user and server that uses your CA will need to have a copy of this file. All parties will rely on the public certificate to ensure that someone is not impersonating a system and performing a Man-in-the-middle attack.

ca.key is the private key that the CA uses to sign certificates for servers and clients. If an attacker gains access to your CA and, in turn, your `ca.key` file, you will need to destroy your CA. This is why your `ca.key` file should only be on your CA machine and that, ideally, your CA machine should be kept offline when not signing certificate requests as an extra security measure.

With that, your CA is in place and it is ready to be used to sign certificate requests, and to revoke certificates.

Generating a CSR

Before a Certificate can be signed, it needs to be created, and an appropriate Certificate Signing Request (CSR) created.

There are many ways to do this, but one of the easiest is using OpenSSL. OpenSSL has a very full features and complex command line. Fortunately there is a an [Easy CSR](#) website provided by Digicert. This allows you to fill in a couple of fields and it will automatically generate an appropriate OpenSSL command line to build the certificate you need.



For this tut its probably best to create a directory called `certs` and then a sub-directory for each certificate you want to generate a CSR for. This keeps all the appropriate files in one place. In a **production** environment, the CSR is usually generated on a secure system, or on the system using the certificate so that the private portion of the key is never disclosed

NOTE: The common name used is either an email address in the case of a user certificate or the FQDN (or even IP) of the system.

An example command line could be:

```
openssl req -new -newkey rsa:2048 -nodes -out an_otherstud_noroff_no.csr
-keyout an_otherstud_noroff_no.key
-subj "/C=NO/ST=Agder/L=Kristiansand/O=CNDDemo/OU=User/CN=an.other@stud.noroff.no"
```

The `.key` file is the private key. The resulting `.csr` file needs to be put somewhere where it can be accessed by the `easy-rsa` scripts. the CSR contains the public key in an appropriate format to be accessed by the CA.

¹A lack of randomness or *entropy* can lead to *BadThings™* when cryptographic operations occur. A prime example of this was the Debian OpenSSL debacle in the mid 2000's. You can read more about it [here](#) and [here](#)



Certificate formats

The PEM format is the most common format used for certificates. Extensions used for PEM certificates are cer, crt, and pem. They are Base64 encoded ASCII files. The DER format is the binary form of the certificate. DER formatted certificates **do not** contain the "BEGIN CERTIFICATE/END CERTIFICATE" statements. DER formatted certificates most often use the '.der' extension.

[Digicert](#) has some very nice examples of using openssl to convert between formats



Generating Certificates

You should generate certificates for the following entities we will be working with:

1. Apache Webserver
2. Mysql DB Server
3. 2x user certificates - one for your @stud.noroff.no address and another for your server backup user - eg recovery@server

Name the CSR files appropriately and then proceed to the next step.

We will be using the user certificates later in the course.

Signing a Certificate

Before signing the certificate we need to import it into the CA system. The last parameter given is the *shortname* you will use for later processing so choose something meaningful.

```
~/easy-rsa$ ./easyrsa import-req ../webserver.csr webserver
```

Note: using Easy-RSA configuration from: ./vars

Using SSL: openssl OpenSSL 1.1.1f 31 Mar 2020

The request has been successfully imported with a short name of: webserver

You may now use this name to perform signing operations on this request.

With the CSR imported we can now proceed with signing.

```
~/easy-rsa$ ./easyrsa sign-req server webserver
```

Note: using Easy-RSA configuration from: ./vars

Using SSL: openssl OpenSSL 1.1.1f 31 Mar 2020

You are about to sign the following certificate.

Please check over the details shown below for accuracy. Note that this request has not been cryptographically verified. Please be sure it came from a trusted source or that you have verified the request checksum with the sender.

Request subject, to be signed as a server certificate for 1080 days:

```
subject=
  countryName             = NO
  stateOrProvinceName     = Agder
  localityName            = Kristiansand
  organizationName        = CNDDemo
  organizationalUnitName   = Network Security
  commonName              = 192.168.41.1
```

Type the word 'yes' to continue, or any other input to abort.

```

Confirm request details: yes
Using configuration from /home/cnd/easy-rsa/pki/safessl-easyrsa.cnf
Enter pass phrase for /home/cnd/easy-rsa/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName             :PRINTABLE:'NO'
stateOrProvinceName     :ASN.1 12:'Agder'
localityName            :ASN.1 12:'Kristiansand'
organizationName        :ASN.1 12:'CNDDemo'
organizationalUnitName  :ASN.1 12:'Network Security'
commonName              :ASN.1 12:'192.168.41.1'
Certificate is to be certified until Oct 24 13:31:51 2023 GMT (1080 days)

Write out database with 1 new entries
Data Base Updated

Certificate created at: /home/cnd/my-ca/pki/issued/webserver.crt

```



For signing **user** CSR's you need to provide the parameter `client` rather than `server` on the command line.

We can verify the signing by looking at the resulting `webserver.crt` file. Large blocks have been shortened for readability in the example on the next page.

An important things to notice are the signed blocks following Signature Algorithm: `ecdsa-with-SHA512` which is the encrypted hash of the provided public key. The base64 encoded block between the

-----BEGIN CERTIFICATE-----

and

-----END CERTIFICATE-----

contains the binary version of the information.

The `webserver.crt` can now be copied back to where the key was generated, and linked into the server configuration.



Locking your CA in DNS

Over a hundred certificate authorities (CAs) have the power to issue certificates which vouch for the identity of your website. Certificate Authority Authorisation (CAA) is a way for you to restrict issuance to the CAs you actually use so you can minimise your risk from security vulnerabilities in all the others.

This makes use of leveraging functionality of putting certificate signatures and 'permission' for a registrar in published in DNS (and potentially secured via DNSSEC). SSLMate provides a useful webpage to [generate](#) records for inclusion in the DNS zones. This is defined in [RFC6844](#). An example record for Noroff could look like:

```

noroff.no. IN CAA 0 issue "digicert.com"
noroff.no. IN CAA 0 issue "letsencrypt.org"
noroff.no. IN CAA 0 issuewild "letsencrypt.org"
noroff.no. IN CAA 0 iodef "mailto:foo@noroff.no"

```

Example Certificate

```
$ cat /home/cnd/my-ca/pki/issued/webserver.crt
```

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

28:24:d0:b8:53:92:9c:b6:85:c3:76:74:87:0a:58:e8

Signature Algorithm: ecdsa-with-SHA512

Issuer: CN=CNDDemo

Validity

Not Before: Sep 8 13:31:51 2023 GMT

Not After : Aug 24 13:31:51 2026 GMT

Subject: C=NO, ST=Agder, L=Kristiansand, O=CNDDemo, OU=Network Security, CN=192.168.41.1

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

RSA Public-Key: (4096 bit)

Modulus:

00:e8:43:e0:91:0d:c2:87:73:55:0a:dc:dd:15:20:

<lines removed>

c3:11:9b

Exponent: 65537 (0x10001)

X509v3 extensions:

X509v3 Basic Constraints:

CA:FALSE

X509v3 Subject Key Identifier:

9F:54:C3:21:1E:F7:34:2A:B5:F4:55:DC:E3:E8:28:B4:0B:61:08:1F

X509v3 Authority Key Identifier:

keyid:00:BC:E2:FE:62:CA:F1:C7:02:86:61:4E:5D:E0:E3:86:52:36:67:4D

DirName:/CN=CNDDemo

serial:25:A0:AA:8F:88:A1:BE:1C:A7:58:6B:F9:1E:6B:5C:E7:C0:BF:BF:F9

X509v3 Extended Key Usage:

TLS Web Server Authentication

X509v3 Key Usage:

Digital Signature, Key Encipherment

X509v3 Subject Alternative Name:

DNS:192.168.41.1

Signature Algorithm: ecdsa-with-SHA512

30:65:02:31:00:e7:ef:34:df:20:0d:0c:c0:36:67:66:39:b2:

<lines removed>

fc:72:c9:67:62:cf:54:ee:af:89:d2:07:7d

-----BEGIN CERTIFICATE-----

MIIEKTCCA6+gAwIBAgIQKCTQuFOSnLaFw3Z0hwpY6DAKBggqhkJOPQQDBDASMRaw

<lines removed>

/HLJZ2LPV06vidIHfQ==

-----END CERTIFICATE-----

**Really want to use ACME/letsencrypt?****EXTENSION EXERCISE ONLY**

You can still use the ACME/letsencrypt free certificates, provided you have a means of validating the hosts. The two most common ways of doing this are:

1. DNS validation - you place an appropriate entry in the zone file for a domain to prove control over it. This requires you have a domain.
2. Host based validation - the issuing bots check on your server website (80/tcp) to determine if you have a correctly named file - again as a means of proving 'ownership' or control over the server.

If these the second may be the easiest. It does however require some network gymnastics to be able to expose a port on a host inside your network to the internet at large (and all the associated risk need to be considered). This can be done using techniques such as port forwarding - either via NAT or SSH. [John Moran](#) provides some interesting discussion around this.

NOTE: this is done at **your own risk**, and the information is provided here for those that feel confident and comfortable.

2 Apache TLS

Now that we have a functional CA and X.509 certificate generation process, we can implement TLS/SSL encryption on our own web server. The basic functionality of the server should have been set up last week. The following activities upgrade the security of the connection by using strong cryptography so information is protected in transit to and from clients rather than being transmitted in plain text.

Apache configuration

Now that we have our signed certificate and key available, we need to update our Apache configuration to use them. On Ubuntu, you can place new Apache configuration files (they must end in `.conf`) into `/etc/apache2/sites-available/` and they will be loaded the next time the Apache process is reloaded or restarted.

For this tutorial we will create a new minimal configuration file.



If you already have an Apache `<VirtualHost>` set up and just need to add SSL to it, you will likely need to copy over the configuration lines that start with SSL, and switch the `VirtualHost` port from 80/tcp to 443/tcp, the latter being that used by the `https://` protocol.



If you have not already done so you will need to enable the SSL/TLS module in Apache by using the command `sudo a2enmod ssl`. If you do not do this, you will get errors in the logs about `SSL Engine` being unknown, and the Apache software will not know how to interpret the SSL specific directives you are giving it.

Create a new file `demotls.conf` in the `/etc/apache2/sites-available` directory and place the following content in it.

```
<VirtualHost *:443>
    ServerName your_ip
    DocumentRoot /var/www/sitepath

    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/apache-signed.crt
    SSLCertificateKeyFile /etc/ssl/private/apache-signed.key
</VirtualHost>
```

It is important that `/var/www/sitepath` exists as this is where we will place content, and that **your_ip** is defined as this is how you will access the server (and thus how the server knows this config applies to it). Replace `sitepath` with what ever you call your directory (and adjust the command below to match)

Create a placeholder index file

```
$ sudo echo "<h1>CND 2022 TLS test worked</h1>" > /var/www/sitepath/index.html
```

When we have content in place the site can be enabled.

```
$ sudo a2ensite demotls.conf
```

With the site enabled, we can test the configuration for errors using `sudo apache2ctl configtest`. This will likely come back with an error as below complaining about the lack of FQDN. This is because there is no functional DNS for the private IP.

```
apache2: Could not reliably determine the server's fully qualified domain name,
        using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message
Syntax OK
```

Restart the server using `sudo systemctl reload apache2`. At this point you should be able to access your site via **https://**. You will likely have a warning message displayed about a non trusted certificate.



The instructions above are a bare minimum of what is needed. There are multiple online resources as to how to configure Apache appropriately. If you **cannot** get it working with your CA certificate try again using a self-signed certificate. When that is working you should be able to substitute in the appropriate certificate files.

HTTPS Redirects

Once you **have SSL/TLS working**, you can add the following configuration to your `demotls.conf` file. This adds a specific virtual host handling traffic on 80/tcp - http - and configures the server to issue a HTTP/302 redirect message to redirect all traffic to the https site running on 443/tcp.

```
<VirtualHost *:80>
    ServerName your_domain_or_ip
    Redirect / https://your_domain_or_ip/
</VirtualHost>
```

Do not forget to check the configuration² `sudo apache2ctl configtest`

If all is well issue a reload - `sudo systemctl reload apache2`



Why Redirect?

Why would you want to put a redirect such as that above in place? What would the effect of this be. What could be a reason you would not want to do this ?



Checking Your Secure Web server

Confirm that the following work/function as expected:

1. Your Apache server starts correctly with no errors in the logs
2. Your webserver is listening on port 443/tcp
3. You can access the webserver locally using `curl` or `wget` to access **both** the http and https prefixed urls. Remember to check their manual pages for options that may be useful.
4. You can access the secure webserver via your browser on a client system.

Trusting the new CA

Having a CA with an ability to sign certificates is well and good, but we need to anchor trust. To do this the public key of the CA needs to be imported into your system and the browser you use specifically.

To do this you require the public key of your CA `ca.crt`. you should also convert this to DER format if you wish to load it into Windows, rather than a browser such as Chrome. The purpose of importing the CA key is to establish a Root Trust Anchor. In essence we trust this key, therefor we trust any keys it signs.



Recognising the CA

Import your CA certificate into a browser on either your host machine, or on a virtualised client system. Once loaded you are likely to still get a SSL/TLS warning in the browser as its not being accessed via a FQDN. You should however be able to see that the certificate is correctly issued and signed.

This process varies slightly for every browser - look for help for the browser you use online. The following shows the result of importing the key into Google Chrome.

²Apache has a habit of just failing to start if there are **any** errors in its configuration. While this is arguably good for security it can be frustrating.

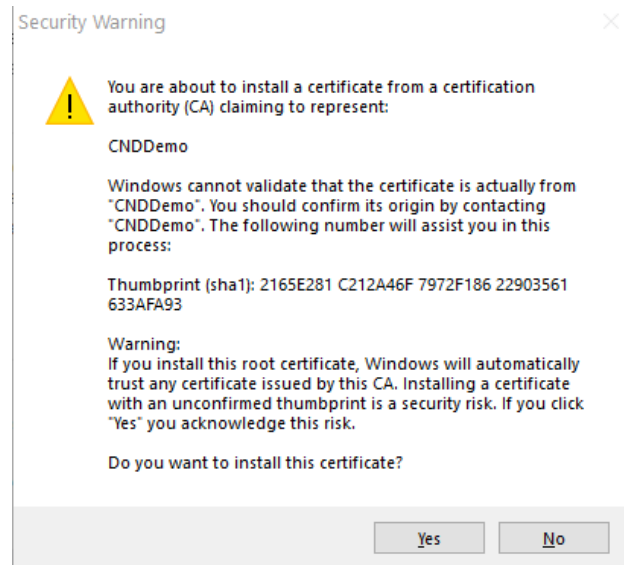


Figure 1: Google Chrome initial import

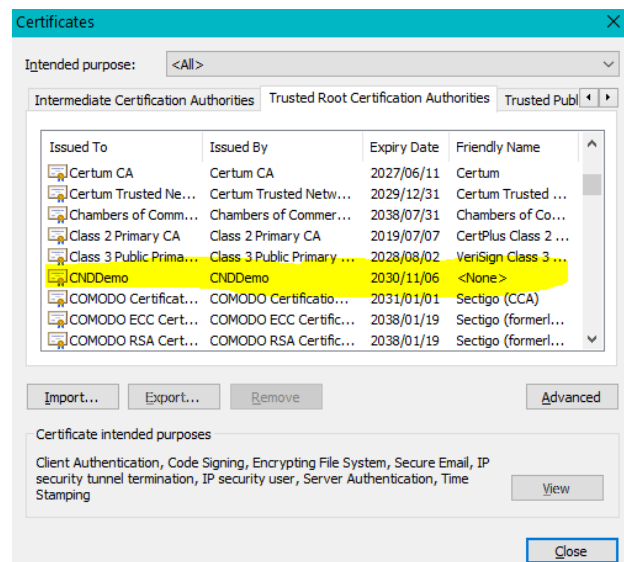


Figure 2: Google Chrome with imported root certificate highlighted



The following resources may be of assistance:

- [How to add a trusted CA certificate to Chrome and Firefox](#)
- [Google Chrome: Import your Personal ID certificate](#)



You do not have to get this trust relationship working, but failing to do so will likely result in you get browser security warnings similar to that below **every time** you access your site.

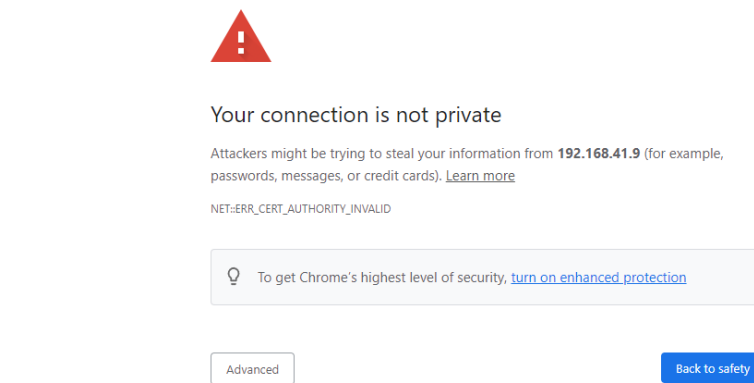


Figure 3: Google Chrome INVALID CA warning

**More on understanding SSL/TLS**

[Delinea](#) provides a good series of articles on the operation of SSL/TLS. This may be worth dipping into if you need to get more understanding.

1. [SSL beyond the basics](#)
2. [Ciphers](#)
3. [Certificates](#)
4. [Strict Transport Security](#)

**TLS in the wild**

Outside of this tutorial where you have a host with a public IP and FQDN, and do not need to full setup of a commercially signed Certificate, you may find the following article of interest. This implements a certificate in Apache using `mod_md` which enable certificate provisioning via the ACME protocol as used by Let's Encrypt free TLS/SSL service. This article explains how to install, set up and configure Apache with the `mod_md` module to secure communications using a Lets Encrypt certificate.

[How To Secure Apache with `mod_md` Let's Encrypt on Ubuntu 20.04 LTS](#)

**Legacy Support**

A common problem one has to deal with 'in the real world' is that of legacy systems. While it would be great to always have all system up to date and compliant with standards this is often not possible. One of the challenges that may arise is how to support older clients (for example older scientific or medical equipment) that does not necessarily support the latest standards. At the same time one does not want to expose 'weak' cipher systems to the Internet at large.

What do you think some ways are of provisioning this?

3 HTTP Headers

The HTTP standard was designed to be extensible, and has served well (now in its 4th major versioning with the emerging HTTP/3 standard). Almost all servers support the venerable HTTP/1.1 protocol with an increasing number supporting HTTP/2. A large part of this extensibility is due to being able to add headers to messages transmitted between the client and the server. Both parties use these in their communication. Headers are effective metadata that are transmitted with each HTTP request or response, that provide additional data about the transaction.

Clients typically send the `Host:` header as part of the HTTP/1.1 Standard. This is needed for Name based virtual hosting³. Servers return a number with every transaction and they provide a large degree of flexibility. Headers form a core part of the operation and smooth negotiation of content between clients and web servers. A list of [common headers](#) are part of the HTTP specification.

Server Fingerprinting

As you have seen when doing undertaking fingerprinting during penetration testing in your second year, webserver headers can reveal a lot about your target. To mask detailed information from Server header, edit Apache's main configuration file. Open the file and add following entries at the end.

```
ServerTokens Full
ServerSignature Off
```

You will need to restart the web server with the command:

```
$ sudo systemctl restart apache2
```

You should see that the Server signature is no longer appended to error pages. When looking at the headers returned by the server the Server versioning is still present (as expected since we have set `ServerTokens Full`).



You can easily check headers by issuing a manual HTTP request doing the following:

- Use `telnet` (or `netcat`) to create a TCP session to the server:
`telnet localhost 80`
- Issue the HTTP request `HEAD / HTTP/1.0` followed by two empty lines (hit enter twice)

```
cnd@cndtut:/etc/apache2$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Sun, 08 Nov 2020 15:15:54 GMT
Server: Apache/2.4.41 (Ubuntu)
Last-Modified: Sat, 24 Oct 2020 10:45:40 GMT
ETag: "2aa6-5b26865efc867"
Accept-Ranges: bytes
Content-Length: 10918
Vary: Accept-Encoding
Connection: close
Content-Type: text/html
```

³[Name-based virtual](#) hosting allows the hosting of multiple website/hosts/domains to be run off a single IP address.



Manual HTTP requests

The examples above and following both make use of manual HTTP requests. Knowing how to do this is a good skill to have, and it applies to many other text based protocols, such as IMAP, POP, WHOIS and SMTP. Other tools you can use are:

- `nc (netcat)`⁴
- `openssl s_client`.

While `nc` is very similar to using `telnet`. To access encrypted versions of these services you should use `openssl s_client` which handles all the crypto negotiations and providing you with a means to interact within that secure tunnel. It can also provide a wealth of information about the ciphers used. As per usual consult the manual page.

Reducing Exposure

Now re-edit the `apache.conf` configuration file and change the line you added from `ServerTokens Full` to `ServerTokens Prod`. Restart the Apache webserver using:

```
$ sudo systemctl restart apache2
```

You can now check the headers and should see something similar to the following:

```
cnd@cndtut:/etc/apache2$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^['.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Sun, 08 Nov 2020 15:16:54 GMT
Server: Apache
Last-Modified: Sat, 24 Oct 2020 10:45:40 GMT
ETag: "2aa6-5b26865efc867"
Accept-Ranges: bytes
Content-Length: 10918
Vary: Accept-Encoding
Connection: close
Content-Type: text/html
```



Note that the line `Server: Apache` no longer discloses the OS or any versioning information. If it displays anything different go back and check your configuration and that you have restarted the Apache server after you saved.

⁴The PE netcat binary almost always gets flagged as malware on windows systems

Removing Headers



Altering headers requires the use of `mod_headers`. `mod_headers` should be enabled by default in Apache, however, you may want to ensure it's enabled by running the following:

```
$ sudo a2enmod headers
$ sudo systemctl restart apache2
```



For all of these you can verify the existence of the header via a manual request, browser plugin, or via snooping traffic (not so easy if you are using HTTPS however). If it displays anything different go back and check your configuration and that you have restarted the Apache server after you saved.

Entity Tags

Entity tags (ETags) are a mechanism that web servers and browsers use to determine whether the component in the browser's cache matches the one on the origin server. ETag is a *validator* which can be used instead of, or in addition to, the Last-Modified header. By sending an ETag, the server promises that the content is not changed until the ETag changes for a specific resource.

The problem with the ETags is that they are generated with attributes that make them unique to a server. By default, Apache will generate an Etag based on the file's *inode*⁵ number, *last-modified date*, and *size*. So, if you have one file on multiple servers with same file size, permissions, timestamp, etc., even after that their ETag will not be the same as they can't have the same *inode* number. While there is not yet a general consensus around the security impact of these headers, it is felt that they should be removed where multiple servers are used in a cluster, or where there is a chance users could have direct access to a file-system.

```
Header unset ETag
FileETag none
```

An alternate approach is to configure the ETag generation mechanism to not include the system inode information and rather just use a combination of file size and its modification time (mtime):

```
FileETag MTime Size
```



This is the type of security configuration that may **not** be appropriate in all contexts. It is important that you understand the environment you are working in and the context in which you are implementing changes. Testing is always important.



Can you find any details on why Etag could be considered a security issue? Discuss with others in your class, and keep notes of the outcome.

PHP Security Headers

While we have yet to specifically address PHP security, it makes sense while we are tightening up to also hide PHP version which is disclosed by the X-Powered-By header. To do that, include following line inside your `/etc/apache2/apache.conf` file.

```
Header unset X-Powered-By
```

After restarting check to see if it is present. This is usually only shown when you have a PHP script executing (so for example try access Wordpress and check that result).

⁵An *inode* (index node) is a data structure in a Unix-style file system that describes a file-system object such as a file or a directory. Each *inode* stores the attributes and disk block locations of the object's data. See [Wikipedia](#) for more information.

4 HTTP Security Headers

As the web has matured and evolved from a place for scientists to share work, and a global repository for [cat pictures](#) to what has rapidly become considered as [Critical Infrastructure](#) security has become increasingly important. While there have been changes made to protocols and encryption standards, headers have provided a means to extend and improve the security of operation without having to re-draft a protocol specification every time.

Working groups have defined (under the auspices of the [W3C](#)⁶) a number of [non-standard](#) headers which can be used to specify certain aspects around security. Despite being non-standard - in other words are not defined in the existing standards - they are very commonly used. Many of the common security headers make use of the so-called X-notation. The HTTP standards define any headers starting with X- as experimental (although this has tended to be ignored and seen rather as header that are not part of the formal specification).



Will it work?

Sites such as [Can I use?](#) provide a convenient way of seeing what compatibility exists for various web browsers (and some web libraries). It is worth checking this when working in Testing and productions to verify if clients in your organisation will be able to support new features you may be relying on.

If you carefully inspect HTTP responses coming from Web Servers such as Google or Facebook or any other appropriately configured server, you might see several non-http-standard headers contained in the response received by the client. An example from Google is:

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Date: Sun, 08 Nov 2020 15:56:35 GMT
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Expires: Sun, 08 Nov 2020 15:56:35 GMT
Cache-Control: private
```

The two notable one as far as this task is concerned are X-XSS-Protection and X-Frame-Options

Some of the others commonly seen, and discussed in lectures are:

- X-Frame-Options
- X-XSS-Protection
- Strict-Transport-Security
- X-Content-Type-Options
- X-Content-Security-Policy/X-WebKit-CSP

These headers are sent by the web server to trigger browser security mechanisms which browsers use to prevent certain Web Application attacks. You can find more information on these headers at this [OWASP](#) page. If not configured manually, these headers are not sent by Apache server and hence browser security mechanisms are not activated.

⁶The [World Wide Web Consortium \(W3C\)](#) serves much the same purpose for Web Standards as the [Internet Engineering Task Force \(IETF\)](#) does for Networking standards through the [RFC](#) program.

Enabling Header Support

To add these to the server response header using Apache, add the following line inside any of the the `<Directory>`, `<Location>`, `<Files>` or `<VirtualHost>` directives (sections) of your server configuration to allow for very precise targeting. These directives are usually located in an included `*.conf` file or your primary configuration file `apache2.conf`, or within a `.htaccess` file within the webroot.



Altering headers requires the use of `mod_headers`. `mod_headers` should be enabled by default in Apache, however, you may want to ensure it's enabled by running the following:

```
$ sudo a2enmod headers
$ sudo systemctl restart apache2
```

Enabling Headers

- X-XSS-Protection
- X-Frame-Options
- X-Content-Type-Options
- HTTP Strict-Transport-Security
- X-Content-Security-Policy/X-WebKit-CSP/Content-Security-Policy

These headers can be enabled in Apache by adding a line of the following format to the apache config file `/etc/apache2/apache2.conf` if you wish to apply to the whole system or one of the `<Directory>`, `<Location>`, `<Files>` or `<VirtualHost>` sections of your server config if you require more flexibility.

```
Header set Headername <options>
```



Make it easier..

You can use the `curl` utility to do your testing rather than manual requests. This is a lot more convenient and makes scripting possible. Refer to the `curl` man page for details of how to show headers only.

In Python the `urllib` provides a powerful interface for managing low-level communications with web hosts. There are a number of other libraries that extend and wrap this functionality in possibly easier to use interfaces.

X-XSS-Protection

The X-XSS-Protection header is designed to enable the cross-site scripting (XSS) filter built into modern web browsers. This is usually enabled by default, but using it will enforce it. It is supported by Internet Explorer 8+, and Safari. Edge, Firefox and Chrome have dropped support in their most recent releases. The recommended configuration is to set this header to the following value, which will enable the XSS protection and instruct the browser to block the response in the event that a malicious script has been inserted from user input, instead of sanitising.

```
X-XSS-Protection: 1; mode=block
```



Further documentation, including syntax and the meaning of various values is available on the [Mozilla X-XSS-Protection](#)



reflected-xss directive.

An important thing to keep in mind is that the X-XSS-Protection header is pretty much being replaced with the new Content Security Policy (CSP) reflected-xss directive. The reflected-xss directive instructs a user agent to activate or deactivate any heuristics used to filter or block reflected cross-site scripting attacks. Valid values are allow, block, and filter. This directive is not supported in the <meta> element. While this increasingly [supported](#) in browsers, it is still recommended to use the X-XSS-Protection header for legacy support. Ideally you should use both the X-XSS-Protection and reflected-xss together.

X-Frame-Options

X-Frame-Options (XFO), is an HTTP response header, also referred to as an HTTP security header, which has been around since 2008. In 2013 it was officially published as [RFC 7034](#), but is not an internet standard. This header tells your browser how to behave when handling your site's content. The main reason for its inception was to provide clickjacking protection by not allowing rendering of a page in a frame. This can include rendering of a page in a <frame>, <iframe>, or <object>.

IFRAMES are used to embed and isolate third party content into a website. Examples of things that use IFRAMES might include social media sharing buttons, Google Maps, video players, audio players, third party advertising, and even some *OAuth* implementations.

```
X-Frame-Options: SAMEORIGIN
```



Further documentation, including syntax and the meaning of various values is available on the [Mozilla X-Frame-Options](#)

X-Content-Type-Options

This header is set by the server to indicate that the MIME types advertised in the Content-Type headers should not be changed and be followed. This is a way to opt out of MIME type sniffing, or, in other words, to say that the MIME types are deliberately configured by the server and should be respected.

Site security testers usually expect this header to be set.

```
X-Content-Type-Options: nosniff
```

The Content-Type is usually set by the server based on file extension, explicitly by an application script or on occasion by doing its own determination of content MIME type (possibly using file magic⁷). An example of a standard HTML page being served up is:

```
Content-Type: text/html; charset=utf-8
```

⁷File magic is often what is referred to by using the `file` command on unix like systems. This uses a collection of [signatures](#) to determine what type a file is.



Further documentation , including syntax and the meaning of various values is available on the [Mozilla X-Content-Type-Options](#)

HTTP Strict Transport Security (HSTS)

NOTE: Before attempting this component, makes sure you have enabled TLS/SSL for your server

This response header is set by the server to lets a web site tell client browsers that it should **only** be accessed using HTTPS, instead of using HTTP. This ensures that compliant browsers cannot downgrade and submit (or retrieve) information over plain text link.

The first time your site is accessed using HTTPS and it returns the Strict-Transport-Security header, the browser records this information, so that future attempts to load the site using HTTP will automatically use HTTPS instead.

When the expiration time specified by the Strict-Transport-Security header elapses, the next attempt to load the site via HTTP will proceed as normal instead of automatically using HTTPS.

Whenever the Strict-Transport-Security header is delivered to the browser, it will update the expiration time for that site, so sites can refresh this information and prevent the timeout from expiring.

```
Strict-Transport-Security: max-age=3600
```

This directive should be used in conjunction with the Rewrite directives from `mod_rewrite` to redirect traffic from `http://` to `https://`. A similar header (but sent by the client) is [Upgrade-Insecure-Requests](#) which indicates that the client wishes to communicate securely.



The Strict-Transport-Security header is **ignored** by the browser when your site is accessed using HTTP; this is because an attacker may intercept HTTP connections and inject the header or remove it. When your site is accessed over HTTPS with no certificate errors, the browser knows your site is HTTPS capable and will honour the Strict-Transport-Security header.



Further documentation , including syntax and the meaning of various values is available on the [Mozilla X-Content-Type-Options](#) page.



HTTP Public Key Pinning (HPKP)

Publishing a fingerprint/hash of a trusted public key is a means of defending against or at least mitigating via warning *Man in the Middle* attack HTTP Public Key Pinning has been deprecated and removed in favour of [Certificate Transparency](#) and [Expect-CT](#). Two options that can be implemented are:

- **Public-Key-Pins**
Associates a specific cryptographic public key with a certain web server to decrease the risk of MITM attacks with forged certificates.
- **Public-Key-Pins-Report-Only**
Sends reports to the report-uri specified in the header and does still allow clients to connect to the server even if the pinning is violated

Content-Security-Policy

Previously implemented as X-Content-Security-Policy or X-WebKit-CSP, this header allows web site administrators to control resources the user agent is allowed to load for a given page. With a few exceptions, policies mostly involve specifying server origins and script endpoints. This helps defend against cross-site scripting attacks (XSS).

As one of the newer security features adopted it has a very rich syntax and options set. These are discussed in the overview [documentation](#). An example which disables in-line evaluation and only allows loading of resources over a secure connection is:

```
Content-Security-Policy: default-src https:
```



Content-Security-Policy Resources


The website content-security-policy.com provides a detailed discussion of the domain of content management via explicit security policies.

NOTE: a complete understanding of CSP and its options is beyond the scope of this course.



Further documentation, including syntax and the meaning of various values is available on the [Mozilla Content-Security-Policy](#) page.



[Scott Helme](#)  runs the [Security Headers](#) website which provides a very detailed scanning and scoring of a websites' implementation of common security headers. The screen shot above shows the result of running a report on nsa.gov, where you can see it only received a C grade. Details as to why are listed on the page.

Work though this site and see what common websites you use (and especially those you think should be secure) score. Try and find an example of grades A-F.

Scan your site now

Scan

☐ Hide results
 ☒ Follow redirects

Security Report Summary

C

Site:	https://www.nsa.gov/
IP Address:	2600:1406:2e00:498::3f78
Report Time:	08 Sep 2022 12:19:52 UTC
Headers:	<div style="display: flex; justify-content: space-between; font-size: 0.9em;"> ✓ X-Frame-Options ✓ X-Content-Type-Options ✓ Strict-Transport-Security ✗ Content-Security-Policy </div> <div style="display: flex; justify-content: space-between; font-size: 0.8em;"> ✗ Referrer-Policy ✗ Permissions-Policy </div>

Figure 4: SecurityHeaders - nsa.gov report

5 Regular Expressions



Whole lot of problems

One people discover the power of regular expressions there is often a temptation to use them to solve all sorts of problems. A very long standing piece of sage advice around these goes as follows:

You have a problem. You use Regex. You now have two problems - Unix proverb.

The cautionary take here is do not jump into using regex, especially if you are trying to cut and paste them from elsewhere. They are an incredibly powerful tool, and an ability to *read* them with some understanding will serve you very well in any kind of future operations or Cyber related career path.

These are an excellent example of the kind of information you should build up in a list of useful things to have. They are not the kind of thing to remember by heart, it is more important to know what you are looking for and where to find it. Make use of the very good online tools which help highlight what is being matched. This is often much easier than banging away at the command line or program and wondering why things are not working. Another important factor to remember there are often multiple ways to do match a given target, these may all work, but some may be more efficient than others. For the purposes of what we are needing, work not speed is important.

There are a wide selection of applications across operating systems for doing local tests on a block of text and your regular expression.

As with the text processing done earlier in the course do not try get it all working at once. Build up your solution.

Having read this, proceed to the next page to learn more.



This task is probably easiest done using the browser based tools. You should however ensure that you can use regular expressions at the command lines especially with `grep`. Take note of the guide in Appendix C. Much of this appendix is a repetition of the content contained in the pages following, but with examples and sample commands *only* using `grep`

This tutorial teaches the basics you need to know to be able to craft powerful time-saving regular expressions. It starts with the most basic concepts, so that you can follow this tutorial even if you know nothing at all about regular expressions yet.

Regular expressions (regex or regexp) are extremely useful in extracting information from any text by searching for one or more matches of a specific search pattern (i.e. a specific sequence of ASCII or unicode characters). Most programming and scripting languages support these, although the specific syntax may differ (check the docs). We will be using one of the most popular formats the so-called PCRE syntax. This is very much based on the regular expression syntax that was developed for the Perl Programming language. The PCRE library (or compatible implementations) exist in a very wide variety of software. Even were the syntax is slightly different you can usually work out what is going on with a little effort.

Fields of application range from validation to parsing/replacing strings, passing through translating data to other formats and web scraping. One of the most interesting features is that once you have learned the syntax, you can actually use this tool in (almost) all programming languages and in a number of different services (such as Apache) with the slightest distinctions about the support of the most advanced features and syntax versions supported by the engines.

Getting your head around regular expressions will make your life much easier as we start using them as part of securing the apache webserver using `mod_rewrite` and `mod_security`. Regex is also invaluable for searching through logs when you don't know what the specific value is but you know what the pattern, or general structure of what you are looking for is. These skills will be of use not only later in the course but as you go out into the working world.

The following sections provide a primer to the concepts and can be used as a reference going forward.

Anchors — `^` and `$`

`^The` matches any string that starts with The

`end$` matches a string that ends with end

`^The end$` exact string match (starts and ends with The end)

`cobra` matches any string that has the text cobra in it

Quantifiers — `*`, `+`, `?` and `{}`

`abc*` matches a string that has ab followed by zero or more c

`abc+` matches a string that has ab followed by one or more c

`abc?` matches a string that has ab followed by zero or one c

`abc{2}` matches a string that has ab followed by 2 c

`abc{2,}` matches a string that has ab followed by 2 or more c

`abc{2,5}` matches a string that has ab followed by 2 up to 5 c

`a(bc)*` matches a string that has a followed by zero or more copies of the sequence bc

`a(bc){2,5}` matches a string that has a followed by 2 up to 5 copies of the sequence bc

OR operator — `|` or `[]`

`a(b—c)` matches a string that has a followed by b or c (and captures b or c)

`a[bc]` same as previous, but without capturing b or c

Capturing in this sense, means keeping track of the match to use elsewhere such as when doing a string substitution.

Character classes — `\d`, `\w`, `\s` and `.`

`\d` matches a single character that is a digit

`\w` matches a word character (alphanumeric character plus underscore)

`\s` matches a whitespace character (includes tabs and line breaks)

`.` matches any character

Use the `.` operator carefully since often class or negated character class (which we'll cover next) are faster and more precise.

`\d`, `\w` and `\s` also present their negations with `\D`, `\W` and `\S` respectively.

For example, `\D` will perform the inverse match with respect to that obtained with `\d`.

`\D` matches a single character that is **not** a digit

`\W` matches a **non-word** character (i.e. **not** alphanumeric character plus underscore)

`\S` matches a **non-whitespace** character (i.e. **not** whitespace, spaces, tabs and line breaks)

Escaping

In order to be taken literally, you must escape the special characters:

`^ . $ [() | * + ? { \`

With a backslash `\` as they have special meaning.

Flags

A regular expression (regex) usually comes in the form of `/abc/`, where the search pattern (`abc`) is delimited by two slash characters `/`.⁸ At the end we can specify a flag with these values (we can also combine them each other):

g (global) does not return after the first match, restarting the subsequent searches from the end of the previous match

m (multi-line) when enabled `^` and `$` will match the start and end of a line, instead of the whole string

i (insensitive) makes the whole expression case-insensitive (for instance `/aBc/i` would match `AbC`)

Whitespace

Whitespace consists of non-printable characters. These are mostly used for formatting, and by their nature do not actually render a character on screen (or paper), but are a very important part of file layout in computing. Regular expressions for the three most important of these are:

`\t` matches tab characters.

`\n` matches the newline character.

`\r` matches the carriage return character.

A very important difference between so called Unix and DOS (and to a large extent Windows text files is how they mark the end of a line. Unix simply uses `\n` whereas DOS formatted text used `\r\n` (better mimicking an old typewriter where the print head went back to the start and then moved down a line). The important takeaway from this is to be aware that sometimes things may not work when trying to match line endings, and to be aware what file types are being used.

In this course we will only use files terminated in the Unix style.



Over to you

Given the information above you can use the online tutorial available at [RegexOne](#).

You should complete at **least** lessons: 1,2,5,7,10.

Ensure you are comfortable with the basics, before applying your skills to the task in the deliverable.

You do not need to work through all the exercises!

RegEx Skill Builder

Now that you have gained a basic understanding of REGULAR EXPRESSIONS, you can go and try your skills against some further online challenges. [Hacker Rank](#) has a selection of challenges ranging from Easy to Hard.



It is recommended that you work through the [Easy](#) challenges in the domains of:

- Introduction
- Character Class
- Repetitions

Extension

To continue to build your skills and confidence once you have completed a few EASY challenges, try stretch to some of the medium ones. This can be done by ticking the MEDIUM difficulty checkbox ☒ on the page.

⁸These delimiters can be changed for example if you are wanting to match slashes. For the vast majority of cases this common format holds true.

Regex Resources

You may find Regex quite daunting. The following Videos may be useful as a more in-depth explanation of how they work along with examples:

- [Regular Expressions \(Regex\) Tutorial: How to Match Any Pattern of Text](#) ⌚ 37:54
- [Learn Regular Expressions In 20 Minutes](#) ⌚ 20:51
- [Learn Regular Expressions \(Regex\) - Crash Course for Beginners](#) ⌚ 45:37



The following other resources may be of use to you:

- <https://www.regular-expressions.info/tutorial.html>
- [Regex cookbook — Top 10 Most wanted regex](#)
- [The Many Uses of Regex](#)
- [Quick-Start: Regex Cheat Sheet](#)
- The [SANS DFIR Regex Cheatsheet](#) by Guy Bruneau, is useful and has tips for some other common tools too.



XKCD 1313 - [Regex Golf](#) © Randal Munroe

6 Deliverable

Real World Regular Expressions

You are required to find/develop appropriate regular expressions for the following cases. In each case example test data is provided that you can use to test your match.

1. Any domain ending in **.com**
MATCH: amazon.com cnn.com
NOMATCH: news.com.au vodacom.co.za
2. A partial URL starting with a specified partial URI e.g. www.example.com/pages/
MATCH: www.example.com/pages/index.html www.example.com/pages/admin/
NOMATCH: www.example.net/pages/index.html example.com/pages/index.html
3. **Any http or https URL**
MATCH: https://www.example.no **and** http://www.example.no
NOMATCH: ftp://www.example.no www.example.no ssh://www.example.no
4. A specific IPv4 address defined as four blocks of numbers eg. aaa.bbb.ccc.ddd
MATCH: 192.129.129.192 **or** 10.23.34.45
NOMATCH: Any address **not specified**
5. **Any valid IPv4 address** defined as four blocks of numbers.
MATCH: 192.129.129.192 **or** 10.23.34.45
NOMATCH: 239.87.257.192 , fe80::20c:29ff:fe58:c7bf , 123.456.678.90
6. IPv4 addresses in the range 192.168.0.0 - 192.168.1.255
MATCH: 192.168.1.3 192.168.0.45
NOMATCH: 192.168.2.14 192.168.123.123
7. An IPv4 addresses in the range 192.168.0.14 - 192.168.0.62
MATCH: 192.168.0.23 192.168.0.62
NOMATCH: 192.168.0.2 192.168.0.128 192.168.0.64
8. Match an IPv4 CIDR range consisting of a dotted quad followed by a / and then a valid subnet size.
MATCH: 192.192.111.64/28 10.0.0.0/8
NOMATCH: 192.129.129.192 192.129.129.192/255.255.255.0 192.129.129.192/64
9. A series of IP addresses within a /8 /16 or /24 netblock. For example, given a block of 10.123.14.67.0/24
MATCH: 10.123.67.14 **and** 10.123.67.32 **and** 10.123.67.0
NOMATCH: 192.129.129.184 **or** 10.123.67.354 **or** 192.129.129.208 **or** 10.123.65.14
10. The hour and minute in a String containing date and time.
 eg. Sep 9 15:08:53 cnd23, 192.129.129.208
 You may assume the string will start at the *beginning* of a line.
MATCH: (relative to the above) 15:08
NOMATCH: Anything else.
11. The process id (pid) of sshd in a system log. For example given the input:


```
Sep 9 13:24:16 cnd23 sshd[1119]: pam_unix(sshd:session): session closed for user cnd
Sep 9 13:25:46 cnd23 sshd[911]: Server listening on 0.0.0.0 port 22.
```


MATCH: 1119 **and** 911
NOMATCH: Any value not the above

12. The process id (pid) of any application in a system log. For example given the input:

```
Sep 9 15:12:22 cnd23 groupadd[2154]: group added to /etc/group: name=plocate, GID=119
Sep 9 15:14:31 cnd23 passwd[2498]: pam_unix(passwd:chauthtok): password changed for passdemo
Sep 9 15:17:01 cnd23 CRON[2501]: pam_unix(cron:session): session opened for user root(uid=0)
Sep 9 15:17:01 cnd23 CRON[2501]: pam_unix(cron:session): session closed for user root
```

MATCH: 2154 and 2498 and 2501

NOMATCH: Any value not the above

For the upcoming engagement where you will need this, you need to ensure you have the following:

- a recorded REGEX pattern for each of the cases above
- are able to use this regex pattern with grep at the command line
- are able to understand and adjust these regex expressions to account for other cases such as a different domain or address



Stuck?

- You can try <https://regexr.com/> which has a very good interactive regex debugger
- You can use grep with a regular expression against a file containing the sample data to try these out.
- Ensure you evaluate **both** positive **and** negative test cases.
- Remember the resources in Appendix C
- Do your best to keep the the **KISS** principle - Keep It Super Simple
- Be very ware of just copying REGEX from online sources (including chat GPT). It is **very important** that you understand how the regular expression is built up so you can modify it if needs be.



Keep Notes

It is important that you keep a record of your regular expressions above. One of the most useful things you can do is start building up a list of regular expressions that you find helpful as you solve problems. Keeping a record of these helps save time, as it is much easier to modify an expression that building one up from scratch.

You will be needing the regular expressions above later in the course

A News

The following are new articles of interest in the last week. These are all relatively short, and you should get in the habit of staying aware of what is going on in the industry at large. The topics discussed here fall in scope for the engagement tasks.



News Week of 11 September 2023

- RESEARCH** [From Caribbean shores to your devices: analysing Cuba ransomware](#)
Cuba mostly targets organisations in the United States, Canada and Europe. The gang has scored a series of resonant attacks on oil companies, financial services, government agencies and healthcare providers. The gang infamously uses complex tactics and techniques to penetrate victim networks, such as exploitation of software vulnerabilities and social engineering.
- ATTACK** [Hackers Exploit Zero-Day Flaw in Software Used by Resorts and Hotels](#)
In the evolving hospitality industry landscape, where vacation rental software has transitioned from luxury to necessity, a growing concern emerges regarding cybersecurity. Of particular interest to financially motivated hackers is credit card information, accounting for a significant 41% of breaches in the hospitality sector, as reported by the Verizon Data Breach Investigations Report.
UPDATE: [MGM Resorts Suffers Cybersecurity Attack, System Outage Reported](#)
- RESEARCH** [Active North Korean campaign targeting security researchers](#)
Threat Analysis Group (TAG) publicly disclosed a campaign from government backed actors in North Korea who used 0-day exploits to target security researchers working on vulnerability research and development. While our analysis of this campaign continues, we are providing an early notification of our initial findings to warn the security research community. We hope this post will remind security researchers that they could be targets of government backed attackers and to stay vigilant of security practices
- DDOS** [Iran's Islamic Propaganda Arm Targeted In Cyberattack](#)
The Iranian Islamic Propaganda Organization has fallen victim to a cyberattack, resulting in temporary disruptions to their online services within Iran attributing the downtime to a Distributed Denial of Service (DDoS) assault.
- POLICY** [The International Criminal Court will now prosecute cyberwar crimes](#)
For years, some cybersecurity defenders and advocates have called for a kind of Geneva Convention for cyberwar, new international laws that would create clear consequences for anyone hacking civilian critical infrastructure. International Criminal Court's will investigate cybercrimes that potentially violate the Rome Statute, the treaty that defines the court's authority to prosecute illegal acts, including war crimes, crimes against humanity, and genocide.

Regarding the use of these in the upcoming engagement the UPDATE in item two **not** included.

B Reading

There are **no** prescribed readings this week, but the following may be of interest in expanding your understanding.



System Hardening

The following links may be of interest relating to system hardening:

- [40 Linux Server Hardening Security Tips](#)
- [Awesome Security Hardening](#)
A very good collection of links to a range of other resources around system hardening across a number of platforms and systems.
- [Guidelines for System Hardening](#) from the Australian government (June 2023)
- [System Hardening Overview](#)



Fantastical Bundle

Fantastical has an offer on a bundle of [video training](#) for Cybersecurity and Networking. This includes topics such as WiFi hacking. All nine modules available for approximately 140 NOK.

C Grep Regular Expressions

This appendix provides a series of worked examples demonstrating and exploring the use of `grep` with regular expressions. The principles covered here apply to many similar commandline tools. While they do not show specific solutions to problems they are intended to provide a clear demonstration of the functionality of different elements that make up the pattern matching of regular expressions.

Introduction

Regular expressions are simple statements that help filter data and files. Many Linux commands, such as the `awk` or `sed` commands you have previously encountered. Regular expressions are also used to find and manipulate information. This is especially useful when data is unstructured, or has a variable structure and not in a clear regulated format such as a csv. Regular expressions are at the simplest level made up of two character types:

- Literals, which are standard text characters.
- Special characters or metacharacters which have special meaning unless escaped with a backslash (\).

You have made use of `grep` in its simplest form, using it to find lines in a file that match some term or phrase. What you may not realise is that by doing this you have already made use of a simple regex functionality.

The syntax for the `grep` command includes regular expressions in the following format:

```
grep [regex] [file]
```

You have used this to search for lines in text. For example to find the user `root` in `/etc/passwd` you can use the following:

```
$ grep root /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

The `grep` command offers three regex syntax options:

1. Basic Regular Expression (BRE) which are used by default
2. Extended Regular Expressions (ERE)
3. Perl Compatible Regular Expressions (PCRE)



Command line regex use

Encase regex expressions in single quotes and escape characters to avoid shell interpretation.



The following examples were generated using a default `.bashrc` for a new user on a Ubuntu 22.04 installation. Other systems will likely have differing output. Ensure you can see that the output is similar in that the results returned match the pattern being searched for.

Basic Regex

A problem with basic regex is its not overly specific. To illustrate this, run the following command as a normal unprivileged user.

```
$ grep if ~/.bashrc
# check the window size after each command and, if necessary,
# set variable identifying the chroot you work in (used in the prompt below)
if [ -z "${debian_chroot:-}" ] && [ -r /etc/debian_chroot ]; then
# uncomment for a colored prompt, if the terminal has the capability; turned
if [ -n "$force_color_prompt" ]; then
    if [ -x /usr/bin/tput ] && tput setaf 1 >&/dev/null; then
if [ "$color_prompt" = yes ]; then
if [ -x /usr/bin/dircolors ]; then
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo error)"
"${history|tail -n1|sed -e '\''s/^\[0-9\]\+\s*//;s/[\;&|]\s*alert$/'\''}'"
if [ -f ~/.bash_aliases ]; then
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
    elif [ -f /etc/bash_completion ]; then
```

The regex searches for the character string. The result shows **all** instances where the letter **i** appears followed by an **f** in the `.bashrc` file in your **home directory**. Therefore, the output highlights the following results:

- **if**
- **elif**
- **notify**
- **identifying**

The command returns only those lines where there is a match. The `-w` flag as you should be aware only returns word matches, but this is not always quite enough to isolate what you are looking for.

How to Use Regex With Grep

Regex offers many possibilities to refine searches with `grep`. Below are some common examples explaining the basic syntax and logic. Combine matches to create complex regex statements.

Literal Matches

Literal matches do an exact match for the specified character string. The previous example expression for `if` demonstrates a literal match.

The search is case-sensitive. The following command returns different results from the previous example:

```
grep If .bashrc
```

Note: Add the `-i` or `--ignore-case` option to match all possible case combinations.

To search for multiple words, add quotation marks to define the phrase. If you omitting quotation marks `grep` treats the second word as a file or location.

```
$ grep "if the" ~/.bashrc
# uncomment for a colored prompt, if the terminal has the capability; turned
```



Remember you do not need to memorise all of these, but you should know they exist both to help you understand an expression you find, and to be able to refer back to a guide such as this when you need it. Most people tend to only remember a couple of phrases/methods they use often and for common tasks. For the rest its RTFM.

Anchor Matches

Anchor matches specify the line location in the search. There are two anchor types as you have seen with the general regex syntax:

```
$ grep ^alias ~/.bashrc
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'
alias alert='notify-send --urgency=low -i "${[ $? = 0 ]}&& echo terminal || echo error)" "$(history|tail
```

The search **does not** display lines with tabs or spaces before the word.

To match lines **ending** with the word then in the .bashrc file, run:

```
$ grep then$ .bashrc
if [ -z "${debian_chroot:-}" ] && [ -r /etc/debian_chroot ]; then
if [ -n "$force_color_prompt" ]; then
    if [ -x /usr/bin/tput ] && tput setaf 1 >&/dev/null; then
if [ "$color_prompt" = yes ]; then
if [ -x /usr/bin/dircolors ]; then
if [ -f ~/.bash_aliases ]; then
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
    elif [ -f /etc/bash_completion ]; then
```

Use both anchors to create a regex statement that searches for a single word or statement in a line:

```
$ grep ^esac$ .bashrc
esac
esac
esac
```

Use **only** anchors to find empty lines in a file. The -n option to shows line numbers in the output. These are relative to the *input* file. Without this the result would be just blank.

```
$ grep -n ^$ .bashrc
4:
10:
...
98:
103:
107:
```

Match Any Character

The period (.) regex metacharacter matches **any** character in place of the sign. For example the following looks for any case of an **r** followed by another character then an **o**. The period can be any character, such as a letter, number, sign, or space.

```
$ grep r.o .bashrc
# set variable identifying the chroot you work in (used in the prompt below)
if [ -z "${debian_chroot:-}" ] && [ -r /etc/debian_chroot ]; then
    debian_chroot=$(cat /etc/debian_chroot)
    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\] '
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
    PS1="\[\e]0;${debian_chroot:+($debian_chroot)}\u@\h: \w\a\]$PS1"
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval "$(dircolors -b)"
# colored GCC warnings and errors
#export GCC_COLORS='error=01;31:warning=01;35:note=01;36:caret=01;32:locus=01:quote=01'
alias alert='notify-send --urgency=low -i "${[ $? = 0 ]}&& echo terminal || echo error)" "$(history|tail
```

Add multiple periods to indicate multiple character placeholders, or combine with anchor matching to create a complex regex statement. Try the following. What do you think they will match ?

```
$grep r..t .bashrc
$grep ..t$ .bashrc
```

Bracket Expressions

Bracket expressions are a more complex form of specifying matches. They allow matching multiple characters or a character range at a position. They follow the pattern of square brackets – [and] – which contain a selection of terms to be matches. The selection works on match **any** of the characters within the brackets. For example, to match all lines that contain **and** or **end** in the .bashrc file, use the following pattern:

```
$ grep [ae]nd .bashrc
# append to the history file, don't overwrite it
shopt -s histappend
# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
# check the window size after each command and, if necessary,
# update the values of LINES and COLUMNS.
# match all files and zero or more directories and subdirectories.
# make less more friendly for non-text input files, see lesspipe(1)
# should be on the output of commands, not on the prompt
# (ISO/IEC-6429). (Lack of such support is extremely rare, and such
# a case would tend to support setf rather than setaf.)
# enable color support of ls and also add handy aliases
# colored GCC warnings and errors
# Add an "alert" alias for long running commands. Use like so:
alias alert='notify-send --urgency=low -i "${[ $? = 0 ]} && echo terminal || echo error)"
"${history|tail -n1|sed -e '\''s/^s*[0-9]\+\s*//;s/[:&|]\s*alert$/'\''}'"
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
```

Bracket expressions allow you to invert the matching (so excluding) characters by adding the caret (^) sign. For example, to match everything *except* for **and** or **end**, use:

```
~$ grep [^ae]nd .bashrc
# check the window size after each command and, if necessary,
# off by default to not distract the user: the focus in a terminal window
```

You can see that the remaining **nd** matched window.

You can extend the range of values matched by bracket expressions. To specify a character range by adding a hyphen (-) between the first and final letter. For example, search for all instances of capital letters:

```
$ grep [A-Z] .bashrc
# If not running interactively, don't do anything
# See bash(1) for more options
HISTCONTROL=ignoreboth
# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000
# update the values of LINES and COLUMNS.
# If set, the pattern "*" used in a pathname expansion context will
...
# You may want to put all your additions into a separate file like
# See /usr/share/doc/bash-doc/examples in the bash-doc package.
```

Bracket expressions allow multiple ranges. For example, match all non-letter characters with:

```
grep [^a-zA-Z] .bashrc
```

Bracket expressions can be combined with anchor matching to find all words starting with capital letters:

```
grep ^[A-Z] .bashrc
```


Character Classes

Character classes are shorthand notation that can be used for common ranges. `grep` support the same standard ones used by many other unix utilities. These can be thought of as a predefined functions to *simplify* bracket expressions. They also make expressions a lot more human readable.

Below is a table that outlines some classes and the bracket expression equivalent. The complete list of `grep` character classes is in the `grep` manual.

Syntax	Description	Equivalent
<code>[:alnum:]</code>	All letters and numbers.	<code>"[0-9a-zA-Z]"</code>
<code>[:alpha:]</code>	All letters.	<code>"[a-zA-Z]"</code>
<code>[:blank:]</code>	Spaces and tabs.	<code>[CTRL+V<TAB>]</code>
<code>[:digit:]</code>	Digits 0 to 9.	<code>[0-9]</code>
<code>[:lower:]</code>	Lowercase letters.	<code>[a-z]</code>
<code>[:punct:]</code>	Punctuation	<code>"[^a-zA-Z0-9]"</code>
<code>[:upper:]</code>	Uppercase letters.	<code>[A-Z]</code>
<code>[:xdigit:]</code>	Hexadecimal digits.	<code>"[0-9a-fA-F]"</code>

Quantifiers

Quantifiers are metacharacters that specify the number of appearances. These can be very useful when trying to be more specific, and when matching numbers which of course consist of consecutive digits. The following table shows each `grep` quantifier syntax with a short description.

Syntax	Description
<code>*</code>	Zero or more matches.
<code>?</code>	Zero or one match.
<code>+</code>	One or more matches.
<code>{n}</code>	<code>n</code> matches.
<code>{n,}</code>	<code>n</code> or more matches.
<code>{,m}</code>	Up to <code>m</code> matches.
<code>{n,m}</code>	From <code>n</code> up to <code>m</code> matches.

Of these the first two are the most commonly used. It is important to understand the specific difference and use cases for these. Look at the difference in output between the following two commands.

```
$ grep m?and .bashrc
$ grep m*and .bashrc
```

In this case, the `*` sign matches the letter `m` zero or more times. Therefore "and, mand, mmand" are all matches. The letter `m` repeats any number of times when followed by the `*` sign. To match zero or exactly one match, use the `?` sign. Encase the statement in single quotes and escape the character to avoid interpretation. For example:

Use range quantifiers to specify an *exact* number of repetitions. For example two approaches to search for strings with two vowels are shown below. The output highlights all words with two vowels. Note the difference is that when using the `-E` parameter the curly braces do not need escaping. This applies to other special characters that may get interpreted by the shell (often the biggest source of frustration when debugging why things do not work),

```
grep '[aeiouAEIOU]\{2\}' .bashrc
grep -E '[aeiouAEIOU]{2}' .bashrc
```

Alternation

Alternation allows for defining *alternative* matches. Encase the alternative strings in single quotes and separate each with an escaped pipe character (`|`). The form using extended regex is also shown which enabled you to omit the escape character. For example, to search for the words `bash` or `alias` in the `.bashrc` file, use:

```
grep 'bash\|alias' .bashrc
```

```
grep -E 'bash|alias' .bashrc
```

Grouping

Regular expressions allow grouping patterns into one item. Place the group in escaped parenthesis for regular regex, or use extended mode to skip the escaping. For example, search for the string `bashrc` and make the `rc` characters optional. In both cases the output highlights **all** instances of `bashrc`. Since `rc` is optional, the command *also* matches the word `bash`.

```
$ grep 'bash\(rc\)\?' .bashrc
```

```
$ grep -E 'bash(rc)?' .bashrc
```

Special Backslash Expressions

There are a few unique backslash expressions for advanced *word boundary* matching. These are worth knowing about and can help shorten and improve the readability of your expressions. Below is a table with short examples for each expression type.

`\b` matches word boundary (whitespace around a word)

`\B` matches empty string provided it is not at a word boundary

`\w` matches a word character (alphanumeric character plus underscore)

`\W` matches a **non-word** character

`\s` matches a whitespace character (includes tabs and line breaks)

`\S` matches a **non-whitespace** character (i.e. **not** whitespace, spaces, tabs and line breaks)

These are similar to the general operators introduced earlier.

For example, use `\b` boundaries to locate a word that is **not** a part of another word. Put otherwise, this ensures there is white-space around the pattern. An example expression that locates the words `see` and `set`. The boundary ensures word isolation.

```
$ grep '\bse[et]\b' .bashrc
```

Escaping Meta-Characters

Escaping meta-characters treats special characters as literals. For example, to search for a period (`.`) at the end of a line, escape the meta-character:

```
$ grep '\.$' .bashrc
```

Not allowing character interpretation helps when searching through source code or configuration files.

Application and Server Security

Computer Network Defence - 2023/24

Version 1.1 28/9/2023

We continue to look at how we can go about securing server systems. A focus these week is on the Apparmor framework which can be used to provide a secondary layer of protection and **mandatory enforcement** of policies on applications running on the system. This provides a much finer grained control beyond what is possible using just the filesystem permissions. Apparmor allows for control of what network resources can be accessed by an application or even how much ram can be consumed. While not appropriate for all applications on a system it provides a very flexible means of providing extra security in the case of 'at risk' application, or applications that could potentially be abused. Continuing with the theme of configuration of subsystems Wordpress is revisited, and a simple SMTP client is installed which is used later when we further explore logging and auditing. Finally cryptographic operations are revisited using OpenSSL. A long standing argument against using strong cryptography is that it can be slow and slows things down. The tutorial task explores this and undertakes some basic benchmarking to help you gain a better understanding around the use of both encryption and hashing. both of these are of value later when looking at validation of systems as well as the protection of backups.

The practical tasks this week consist of five distinct groupings:

SYSTEM SECURITY *[Task 1 & 2]* Explore the use of Apparmor as a means of Mandatory access Control on a Linux system. **Read the details around the relevance of these very carefully .**

HARDENING *[Task 3 – 5]* Continue with configuration though the exploration of some security concepts of Databases and Wordpress, relating to lectures, and the configuration of a client to enable the server to send email. **Task 5 has dependencies later in the course and must be completed.**

CRYPTO *[Tasks 6-8]* Explores the performance and application of of crypto functions related to both encryption and secure hashing. Tasks 7 and 8 relate to configuring and evaluating the cryptographic algorithms used on a web server

PREPARATION *[Task 9]* is a brief list of networking and network security topics to ensure you have reviewed and prepared before we start with this aspect of the course.

READING There are five brief news articles to review.

These broad areas can all be worked on independently, with no inter-dependency between the sections. The order in which you approach these three areas is up to you. You are strongly encouraged to switch between tasks if you get stuck or find your concentration drifting. The Deliverable for the week focuses on your development of a TLS auditing script, and undertaking some benchmarking. It also lists as a summary items from other tasks that you should make sure you have kept notes about. You are expected to have these completed prior to the next tutorial. The activities there in build on the concepts covered and skills developed in the preceding tasks, as well as your prior weeks. There is also a guide on what further preparation and knowledge refresh you should undertake during the Reading Week.

There are five brief news articles to review, along with a listing of some security related tools you may find useful.

This tut *may* be updated during the course of the week with additional support material, make sure you have the latest version as shown on Moodle.

Contents

1	Linux AppArmor	1
2	Customising AppArmor	9
3	MySQL Hardening	14
4	WordPress Security	17
5	Email with SSMTP	19
6	Benchmarking Crypto	23
7	TLS Tuning	27
8	Deliverable	31
9	Preparation	33
	Resources	33
A	News	34
B	Tools	35
C	OpenSSL	36

CHANGELOG

1. v1.0 - Initial Release
 2. v1.01 - Removed "AES is disabled" from Crypto deliverable. Added clarification in Task 5 around app passwords.
 3. v1.1 - Added extra debugging detail in Task 3
-

1 Linux AppArmor

This task is **not** in scope for Mandatory engagement activities. There are also no other activities that depend on this (other than Task 2 in this tutorial). It **does** however form part of the examinable content for the course.

AppArmor ("Application Armor") is a Linux kernel security module that allows the system administrator to restrict programs' capabilities with per-program profiles. Profiles can allow capabilities like network access, raw socket access, and the permission to read, write, or execute files on matching paths. AppArmor supplements the traditional Unix discretionary access control (DAC) model by providing mandatory access control (MAC).

In addition to manually creating profiles, AppArmor includes a learning mode, in which profile violations are logged, but not prevented. This log can then be used for generating an AppArmor profile, based on the program's typical behaviour. AppArmor is implemented using the Linux Security Modules (LSM) kernel interface

This task provides an introduction to the use of AppArmor on an Ubuntu Linux system, and the development of some custom security enhancements using it.



Before proceeding make sure you have a backup or snapshot of your system.

Getting Started

This section provides an introduction to the apparmor application suite. Run the following to install some additional utilities:

```
apt install apparmor-utils
```

The presence and activation status of apparmor can be verified using the `aa-status` command. An example output from a default Ubuntu Server install is shown below.

```
root@cnd23:/home/cnd# aa-status
apparmor module is loaded.
32 profiles are loaded.
32 profiles are in enforce mode.
  /snap/snapd/16292/usr/lib/snapd/snap-confine
  /snap/snapd/16292/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
  /snap/snapd/16778/usr/lib/snapd/snap-confine
  /snap/snapd/16778/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
  /usr/bin/man
  /usr/lib/NetworkManager/nm-dhcp-client.action
  /usr/lib/NetworkManager/nm-dhcp-helper
  /usr/lib/connman/scripts/dhclient-script
  /usr/lib/snapd/snap-confine
  /usr/lib/snapd/snap-confine//mount-namespace-capture-helper
  /usr/sbin/mysqld
  /{,usr/}sbin/dhclient
  lsb_release
  man_filter
  man_groff
  nvidia_modprobe
  nvidia_modprobe//kmod
  snap-update-ns.lxd
  snap.lxd.activate
  snap.lxd.benchmark
  snap.lxd.buginfo
  snap.lxd.check-kernel
  snap.lxd.daemon
  snap.lxd.hook.configure
```

```

snap.lxd.hook.install
snap.lxd.hook.remove
snap.lxd.lxc
snap.lxd.lxc-to-lxd
snap.lxd.lxd
snap.lxd.migrate
snap.lxd.user-daemon
tcpdump
0 profiles are in complain mode.
0 profiles are in kill mode.
0 profiles are in unconfined mode.
1 processes have profiles defined.
1 processes are in enforce mode.
  /usr/sbin/mysqld (29865)
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
0 processes are in mixed mode.
0 processes are in kill mode.

```

This listing shows all the binaries/commands/programs for which there exist profiles on the system. A profile must exist for the kernel framework to be able to monitor and control activity. These profiles are contained in `/etc/apparmor.d/`. Not all of them may be running on your system

```

root@cnd23:/home/cnd# ls -l /etc/apparmor.d/
total 80
drwxr-xr-x 2 root root 4096 Aug  9 11:57 abi
drwxr-xr-x 4 root root 4096 Aug  9 11:57 abstractions
drwxr-xr-x 2 root root 4096 Aug  9 11:58 disable
drwxr-xr-x 2 root root 4096 Jun 21 04:46 force-complain
drwxr-xr-x 2 root root 4096 Sep 15 11:53 local
-rw-r--r-- 1 root root 1339 Jun 21 04:46 lsb_release
-rw-r--r-- 1 root root 1189 Jun 21 04:46 nvidia_modprobe
-rw-r--r-- 1 root root 3461 Jun 21 10:38 sbin.dhclient
drwxr-xr-x 5 root root 4096 Aug  9 11:57 tunables
-rw-r--r-- 1 root root 3448 Mar 17 2022 usr.bin.man
-rw-r--r-- 1 root root 1421 Jun 20 2021 usr.bin.tcpdump
-rw-r--r-- 1 root root 28376 Aug  8 09:17 usr.lib.snapd.snap-confine.real
-rw-r--r-- 1 root root 2006 Jul 26 16:14 usr.sbin.mysqld
-rw-r--r-- 1 root root 1592 Nov 16 2021 usr.sbin.rsyslogd

```

You can list running executable files which are not currently confined by an AppArmor profile using the two commands `aa-unconfined` and the more restrictive `aa-unconfined --paranoid`.

`aa-unconfined` will use `netstat(8)`, or `ss(8)` as a backup, to determine which processes have open network sockets and do not have AppArmor profiles loaded into the kernel. An *example* output is shown below. The exact output will depend on what you have running.

```

# aa-unconfined
785 /usr/lib/systemd/systemd-networkd (/lib/systemd/systemd-networkd) not confined
787 /usr/lib/systemd/systemd-resolved (/lib/systemd/systemd-resolved) not confined
844 /usr/sbin/sshd (sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups) not confined
11542 /usr/sbin/apache2 not confined
11544 /usr/sbin/apache2 not confined
11545 /usr/sbin/apache2 not confined
30206 /usr/bin/nc.openbsd not confined

```

This lists process numbers (pid) in the first column and then the full path. You can note that in this case, `netcat` (`/usr/bin/nc.openbsd`) is running as pid 30206.



Finding ports

You can use the following to see what ports are open or listening on a system.

```
netstat -4a | grep LISTEN
tcp 0 0 0.0.0.0:1234 0.0.0.0:* LISTEN
tcp 0 0 localhost:domain 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:ssh 0.0.0.0:* LISTEN
```

You can list processes that have network tcp or udp ports open that are also 'unconfined' using the following command:

```
root@cnd23:/home/cnd# ps auxZ | grep -v '^unconfined'
```

Case Study: Making tcpdump usable

As seen when using `aa-status` and listing the profiles, `/usr/sbin/tcpdump` is one of the binaries that by default has controls in place via AppArmor on an Ubuntu System. A side effect of this is that the default Ubuntu historical policies have been a little restrictive. This has been remedied in the 22.04 release but still serves as a way to demonstrate the capabilities of Apparmor.

Reverting configuration

We need to revert the behaviour to what existed in prior Ubuntu versions. Historically by default a user cannot open any `.cap` files they do not own, only `.pcap`. These are both formats used for recording packet captures, and largely treated as interchangeable and down to preference preference.

We are going to configure this legacy behaviour and then revert to the more modern (and sane configuration).

The following shows you how this can be changed to allow similar functionality for `.cap` files as for `.pcap`. As a privileged user you need to edit `/etc/apparmor.d/usr.bin.tcpdump` and change the two lines from

```
# for -r, -F and -w
/**. [pP] [cC] [aA] [pP] rw,
/**. [cC] [aA] [pP] rw,
```

now commenting out the handling for `.cap` files to

```
# for -r, -F and -w
/**. [pP] [cC] [aA] [pP] rw,
#/**. [cC] [aA] [pP] rw,
```

Reload the Apparmor profile

```
$ sudo aa-enforce /etc/apparmor.d/usr.bin.tcpdump
Setting /etc/apparmor.d/usr.bin.tcpdump to enforce mode.
```

Generate some capture files

As the `root` user or using `sudo` appropriately you will need to generate some network capture files to use in your testing. The following command will capture 100 packets from your primary network interface. replace `ens33` with the name of the network interface on your system as needed. You must save these captures in the home directory of your standard user on the system.

```
tcpdump -ni ens33 -c 100 -w test.cap
```

This command captures 100 packets from the interface `ens33` and writes it to a file called `test.cap`. You may need to generate enough traffic to the VM system for it to exit. You can create another session log in and do a couple of directory listings, access a web page, or send some ping packets. The `tcpdump` command will exit once it has 100 packets recorded.



What interface?

ens33 is the default interface for most ubuntu servers running under VMWare. If this fails you can check what interfaces are present by running `ifconfig` or `ip address`. Look for the interface name that matches the IP address of your server.

After the `test.cap` file can be created, copy it to `test.pcap` using `cp test.cap test.pcap` and finally set the ownership for both to `root:root`. You will need to use `sudo` to do this.

As your **standard user** (in this example `cnd`) you should have something that looks similar to the listing below.



Ensure that the file permissions and ownership are set correctly before proceeding.

```
cnd@cnd23:/home/cnd$ ls -la
total 100
drwxr-x--- 5 cnd      cnd      4096 Sep 22 17:28 .
drwxr-xr-x 4 root     root     4096 Sep 15 15:08 ..
-rw----- 1 cnd      cnd      3715 Sep 22 17:15 .bash_history
-rw-r--r-- 1 cnd      cnd        220 Jan  6  2022 .bash_logout
-rw-r--r-- 1 cnd      cnd     3771 Jan  6  2022 .bashrc
drwx----- 2 cnd      cnd      4096 Aug 25 12:55 .cache
drwx----- 4 cnd      cnd      4096 Sep  1 14:09 .gnupg
-rw----- 1 cnd      cnd        33 Sep 15 14:27 .lessht
-rw-r--r-- 1 cnd      cnd       807 Jan  6  2022 .profile
drwx----- 2 cnd      cnd      4096 Aug 25 12:54 .ssh
-rw-r--r-- 1 cnd      cnd         0 Aug 25 13:02 .sudo_as_admin_successful
-rw-r--r-- 1 tcpdump  tcpdump 20550 Sep 22 17:26 test.cap
-rw-r--r-- 1 root     root    20550 Sep 22 17:28 test.pcap
-rw----- 1 cnd      cnd      5964 Sep 22 16:38 .viminfo
```

Testing tcpdump

With the test data in place, try reading the generated capture files. You can use the command `tcpdump -nqr <file>` to read them, replacing `<file>` with the filename you want to read. More on `tcpdump` can be found in its manual page - `man tcpdump`. It is important that you run these commands as a normal user and **not** use `sudo`. What are your results ?

Sample output should look similar to the following:

```
cnd@cnd23:~$ tcpdump -nqr test.pcap -c 5
reading from file test.pcap, link-type EN10MB (Ethernet), snapshot length 262144
17:26:06.305395 IP 192.168.15.129.22 > 192.168.15.1.55011: tcp 112
17:26:06.306462 IP 192.168.15.129.22 > 192.168.15.1.55011: tcp 80
17:26:06.307015 IP 192.168.15.129.22 > 192.168.15.1.55011: tcp 64
17:26:06.309958 IP 192.168.15.1.55011 > 192.168.15.129.22: tcp 0
17:26:17.462755 IP 192.168.15.1.55230 > 192.168.15.129.22: tcp 0
```

```
cnd@cnd23:~$ tcpdump -nqr test.cap -c 5
tcpdump: test.cap: Permission denied
```

This should clearly show that you are unable to read from the one file despite the permissions being the same, and readable by all users. If you copy the `test.cap` file as a standard user, can you read it. Why do you think so ? This is because the user now owns the file and is thus deemed 'safe'. An example of the listing is shown below.

```
cnd@cnd23:~$ ls -la
total 124
drwxr-x--- 5 cnd      cnd      4096 Sep 22 17:42 .
```

```
drwxr-xr-x 4 root root 4096 Sep 15 15:08 ..
-rw----- 1 cnd cnd 3715 Sep 22 17:15 .bash_history
-rw-r--r-- 1 cnd cnd 220 Jan 6 2022 .bash_logout
-rw-r--r-- 1 cnd cnd 3771 Jan 6 2022 .bashrc
drwx----- 2 cnd cnd 4096 Aug 25 12:55 .cache
drwx----- 4 cnd cnd 4096 Sep 1 14:09 .gnupg
-rw----- 1 cnd cnd 46 Sep 22 17:41 .lessht
-rw-r--r-- 1 cnd cnd 807 Jan 6 2022 .profile
drwx----- 2 cnd cnd 4096 Aug 25 12:54 .ssh
-rw-r--r-- 1 cnd cnd 0 Aug 25 13:02 .sudo_as_admin_successful
-rw-r--r-- 1 cnd cnd 20550 Sep 22 17:42 test2.cap
-rw-r--r-- 1 root root 20550 Sep 22 17:26 test.cap
-rw-r--r-- 1 root root 20550 Sep 22 17:28 test.pcap
-rw----- 1 cnd cnd 5964 Sep 22 16:38 .viminfo
cnd@cnd23:~$ tcpdump -qnr test2.cap -c 5
reading from file test2.cap, link-type EN10MB (Ethernet), snapshot length 262144
17:26:06.305395 IP 192.168.15.129.22 > 192.168.15.1.55011: tcp 112
17:26:06.306462 IP 192.168.15.129.22 > 192.168.15.1.55011: tcp 80
17:26:06.307015 IP 192.168.15.129.22 > 192.168.15.1.55011: tcp 64
17:26:06.309958 IP 192.168.15.1.55011 > 192.168.15.129.22: tcp 0
17:26:17.462755 IP 192.168.15.1.55230 > 192.168.15.129.22: tcp 0
cnd@cnd23:~$
```



AppArmor configuration format

You should read the following resources to help you understand how AppArmor profiles are created:

- Section 5 of the system Manual for `apparmor.d` – `man apparmor.d`
- SuSe [documentation](#) on apparmor profiles.

AppArmor Profile

As seen earlier, the profile for `/usr/sbin/tcpdump` is contained in `/etc/AppArmor.d/usr.sbin.tcpdump`. Looking at this file there is a section of interest which looks to be related to the behaviour we are seeing of preventing access

To confirm the behaviour is related to AppArmor, disable the profile for this command using `aa-disable` and run the tests again.

```
cnd@cnd23:~$ sudo aa-disable /etc/apparmor.d/usr.bin.tcpdump
Disabling /etc/apparmor.d/usr.bin.tcpdump.
cnd@cnd23:~$ tcpdump -qnr test.cap -c 5
reading from file test.cap, link-type EN10MB (Ethernet), snapshot length 262144
17:26:06.305395 IP 192.168.15.129.22 > 192.168.15.1.55011: tcp 112
17:26:06.306462 IP 192.168.15.129.22 > 192.168.15.1.55011: tcp 80
17:26:06.307015 IP 192.168.15.129.22 > 192.168.15.1.55011: tcp 64
17:26:06.309958 IP 192.168.15.1.55011 > 192.168.15.129.22: tcp 0
17:26:17.462755 IP 192.168.15.1.55230 > 192.168.15.129.22: tcp 0
cnd@cnd23:~$
```

Read tests for all three files should now succeed, indicating this **is** an AppArmor issue as suspected.

1. Re-enable the profile but this time using `aa-complain`.
2. The result of this is that any violations of policy will be flagged in the system logs. These can be found in `/var/log/syslog`.
3. Try processing `test.cap` again. this should succeed.
4. If you search the logs you should see entries similar to those below. The first entry shows when the activity was denied, and the second now in complain mode.

```
Sep 22 17:46:44 cnd23 kernel: [ 1834.497284] audit: type=1400 audit(1663868804.079:41):
apparmor="ALLOWED" operation="open" profile="tcpdump" name="/home/cnd/test.cap" pid=1520
comm="tcpdump" requested_mask="r" denied_mask="r" fsuid=1000 ouid=0
```

The key factor here is the entry `denied_mask="r"` in the logs. This indicates that the `tcpdump` is unable to **open** the `test.cap` file for reading as shown by the `r`.

You should now re-edit the `tcpdump` Apparmor profile back to its original state, and reload it using `aa-enforce`. you should be able to read `test.cap` as a normal user despite the file not being owned by the user.



Reloading profiles

Once a profile has been edited, reload the profile in the kernel with `apparmor_parser(8)`:

```
$ sudo apparmor_parser -r /etc/apparmor.d/usr.bin.example
```



Evaluating your AppArmor policy

AppArmor provides additional permission checks to traditional Discretionary Access Controls (DAC).

- DAC is *always* checked in addition to the AppArmor permission checks. As such, AppArmor cannot override DAC to provide **more** access than what would be normally allowed.
- AppArmor normalizes path names. It resolves symlinks and considers each hard link as a different access path.
- Deny rules cannot be overridden by an allow rule.
- Creation of files requires the create permission (implied by w) on the path to be created. Separate rules for writing to the directory of where the file resides are not required.
- Deletion works like creation but requires the delete permission (implied by w).
- Copy requires 'r' of the source with create and write at the destination (implied by w). Move is like copy, but also requires delete at source.

The profile must be loaded **before** an application starts for the confinement to take effect, but policy may be reloaded while the application is running with the rules taking effect immediately. You will want to make sure that you load policy during boot **before** any confined daemons or applications. This is done for you in Ubuntu.



AppArmor Pattern Matching

AppArmor uses a file globbing syntax similar to that used by the bash shell. Globbing is not standard full regular expression syntax instead it uses a few characters known as wild cards. The AppArmor wildcards have slightly *different* semantics than that of bash.

- * - match zero or more characters at the directory level. When looking at the path as a string this will match every character except /. This will match dot files (file names starting with .), excepting the special dot files . and .., if it is placed immediately after the directory / .
- ** - match 0 or more characters over multiple directory levels. This will match dot files (file names starting with .), excepting the special dot files . and .., if it is placed immediately after the directory /
- ? - match a single character that is not /
- { } - alternation - a comma separated list of alternate strings that can be matched. An empty string is allowed and means no string is a viable alternative
- [] - character class
- [^] - inverted character class



AppArmor language syntax

< *name* > denotes a subpattern (rule). That is matched against.

< *name* >:= begins the definite of a subpattern (rule).

' ' single quotes are used to denote literal text

() group rules and text together

[] square bracket denote the enclosed pattern is optional, that is can appear 0 or 1 time

* a trailing * on a pattern indicate the pattern may appear 0 or more times

+ a trailing + on a pattern indicate the pattern may appear 1 or more times

| is used to separate alternate subpatterns within an expression



Given your knowledge of regular expressions and the details above, consider the section highlighted below in the AppArmor profile for `tcpdump` above.

- What is the profile doing?
- What file names will match the default permissions in the profile?
- What does the line mentioning `snort` permit for a standard user?
- If you copy `test.cap` and `test.pcap` to `/tmp` are you able to read them as a standard user?
- How do you verify this when running in enforce mode with `aa-enforce`?

```
# for -F and -w
audit deny @{HOME}/.* mrwkl,
audit deny @{HOME}/.* / rw,
audit deny @{HOME}/.*/** mrwkl,
audit deny @{HOME}/bin/ rw,
audit deny @{HOME}/bin/** mrwkl,
owner @{HOME}/ r,
owner @{HOME}/** rw,

# for -r, -F and -w
/**.[pP][cC][aA][pP] rw,
/**.[cC][aA][pP] rw,

# for convenience with -r (ie, read pcap files from other sources)
/var/log/snort/*log* r,

/usr/bin/tcpdump mr,
```



At the completion of this task, ensure that you have reverted the configuration back to the default you started with.

2 Customising AppArmor



Extension Task This task is not required, and does **not** form part of examinable content and is provided for those that want to do some more work with Apparmor.

You should use the previous task along with appropriate readings to complete the following task.

As a demonstration of your newly acquired skills in using AppArmor as a tool to control aspects of system security, you need to develop an appropriate AppArmor profile for the following script.bin

```
#!/bin/bash
echo "CND AppArmour Demonstration"

echo "Building data and creating file"
echo $hostname > hostname.tmp && cat hostname.tmp | /usr/bin/md5sum > hostname.md5
cp hostname.md5 hostname2.md5
echo "Erasing File"
rm hostname.tmp hostname2.md5

echo "Touching /tmp"
cp hostname.md5 /tmp/hostname.md5
echo "Reading /tmp"
cat /tmp/hostname.md5
```

1. Save this script and take note of the **full** path where it is located. For example if I save the script in the cnd user's home directly under bin. The full path will be `/home/cnd/bin/example.sh`.
2. Remember to set it executable.
3. Create a test directory

```
cnd@cnd22:~$ mkdir /home/cnd/testing
cnd@cnd22:~$ cd /home/cnd/testing
```

4. Execute your script and check the output

```
cnd@cnd22:~/testing$ /home/cnd/bin/example.sh
CND AppArmour Demonstration
Building data and creating file
Erasing File
Touching /tmp
Reading /tmp
68b329da9893e34099c7d8ad5cb9c940 -
cnd@cnd22:~/testing$ ls -la
total 12
drwxrwxr-x 2 cnd cnd 4096 Sep 22 18:07 .
drwxr-x--- 7 cnd cnd 4096 Sep 22 18:04 ..
-rw-rw-r-- 1 cnd cnd 36 Sep 22 18:07 hostname.md5
$ ls -lad /tmp/hostname.md5
-rw-rw-r-- 1 cnd cnd 36 Sep 22 18:10 /tmp/hostname.md5
```

You are now ready to start creating a suitable Apparmor profile.

Creating a custom profile

1. The easiest way to start is by using aa-easyprofile to generate a starting template.

```
$ aa-easyprof /home/cnd/bin/example.sh
# vim:syntax=apparmor
# AppArmor policy for example.sh
# ###AUTHOR###
# ###COPYRIGHT###
# ###COMMENT###

#include <tunables/global>

# No template variables specified

"/home/cnd/bin/example.sh" {
    #include <abstractions/base>

    # No abstractions specified

    # No policy groups specified

    # No read paths specified

    # No write paths specified
}
```

2. This is worth writing to a skeleton in the profiles directory. Remember to use sudo

```
sudo aa-easyprof /home/cnd/bin/example.sh > /tmp/aaprofile
sudo cp /tmp/aaprofile /etc/apparmor.d/home.cnd.bin.example.sh
```

3. Now load the profile into the kernel using apparmor_parser

4. Execute the script as your normal user.

```
cnd@cnd22:~/testing$ /home/cnd/bin/example.sh
/bin/bash: /home/cnd/bin/example.sh: Permission denied
```

5. This fails since the profile has no permissions loaded.

6. The logs will need to be investigated. You should see something similar to the following:

```
Sep 22 18:20:07 cnd22 kernel: [ 3837.433207] audit: type=1400 audit(1663870807.009:44):
apparmor="STATUS" operation="profile_load" profile="unconfined"
name="/home/cnd/bin/example.sh" pid=1675 comm="apparmor_parser"
Sep 22 18:20:55 cnd22 kernel: [ 3886.405781] audit: type=1400 audit(1663870855.981:45):
apparmor="DENIED" operation="open" profile="/home/cnd/bin/example.sh"
name="/dev/tty" pid=1678 comm="example.sh" requested_mask="wr" denied_mask="wr" fsuid=1000 ouid=0
Sep 22 18:20:55 cnd22 kernel: [ 3886.405803] audit: type=1400 audit(1663870855.981:46):
apparmor="DENIED" operation="open" profile="/home/cnd/bin/example.sh"
name="/dev/pts/2" pid=1678 comm="example.sh" requested_mask="wr" denied_mask="wr"
fsuid=1000 ouid=1000
Sep 22 18:20:55 cnd22 kernel: [ 3886.406773] audit: type=1400 audit(1663870855.981:47):
apparmor="DENIED" operation="open" profile="/home/cnd/bin/example.sh"
name="/home/cnd/bin/example.sh" pid=1678 comm="example.sh" requested_mask="r" denied_mask="r"
fsuid=1000 ouid=1000
Sep 22 18:21:08 cnd22 kernel: [ 3898.805477] audit: type=1400 audit(1663870868.381:48):
apparmor="DENIED" operation="open" profile="/home/cnd/bin/example.sh"
name="/dev/tty" pid=1680 comm="example.sh" requested_mask="wr" denied_mask="wr"
fsuid=1000 ouid=0
```

```
Sep 22 18:21:08 cnd22 kernel: [ 3898.805703] audit: type=1400 audit(1663870868.381:49):
apparmor="DENIED" operation="open" profile="/home/cnd/bin/example.sh"
name="/dev/pts/2" pid=1680 comm="example.sh" requested_mask="wr" denied_mask="wr" fsuid=1000
ouid=1000
Sep 22 18:21:08 cnd22 kernel: [ 3898.807116] audit: type=1400 audit(1663870868.385:50):
apparmor="DENIED" operation="open" profile="/home/cnd/bin/example.sh"
name="/home/cnd/bin/example.sh" pid=1680 comm="example.sh" requested_mask="r" denied_mask="r"
fsuid=1000 ouid=1000
```

7. This shows us what was being attempted and what was **denied**. Look carefully at the lines with "apparmor="DENIED"



Another way to view AppArmor denials is by using the aa-notify tool. aa-notify is a very simple program that will report any new AppArmor denials by consulting /var/log/syslog (or /var/log/audit/audit.log if auditd is installed). For example:
/usr/bin/aa-notify -s 1 -v
will show any AppArmor denials within the last day.

8. An easy way to generate the profile is to use the aa-logprof tool.
9. Reload the profile using aa-complain
10. Run the script again. This time it should succeed, and generate a number of ALLOW entries in the logs.
11. Review the logs. Each entry shows what the script was trying to do.
12. aa-logprof can now be run. This will read back entries and ask you a series of questions.

```
cnd@cnd22:~/testing$ sudo aa-logprof
Updating AppArmor profiles in /etc/apparmor.d.
Reading log entries from /var/log/syslog.
```

```
Profile: /home/cnd/bin/example.sh
Execute: /usr/bin/cat
Severity: unknown
```

```
(I)nherit / (C)hild / (N)amed / (X)ix On / (D)eny / Abo(r)t / (F)inish
```

13. you can use (I)nherit to allow the standard permissions for a file, or (D)eny to stop an action. See the Apparmor documentation for more details.
14. When files or devices (such as tty - the terminal) you can use (A)llow or (D)eny
15. at this point use a combination of (I)nherit and (A)llow to generate the base profile and then save.

16. The base profile has now been fileed out a little and should look like:

```
# Last Modified: Thu Sep 22 19:20:47 2022
include <tunables/global>

# vim:syntax=apparmor
# AppArmor policy for example.sh
# ###AUTHOR###
# ###COPYRIGHT###
# ###COMMENT###
# No template variables specified

/home/cnd/bin/example.sh flags=(complain) {
    include <abstractions/base>
    include <abstractions/consoles>

    /etc/ld.so.cache r,
    /usr/bin/cat mrix,
    /usr/bin/cp mrix,
    /usr/bin/md5sum mrix,
    /usr/bin/rm mrix,
    owner /home/*/bin/example.sh r,
    owner /home/*/testing/hostname.md5 w,
    owner /home/*/testing/hostname.tmp rw,
}
```

17. With this base profile in place you are now ready to do some further exploration to adjust the profile to the goals below.
18. Re-execute the script. What doesnt work ? Why is this ?
19. When editing you should take this opportunity to touch up the AUTHOR, COPYRIGHT, and COMMENT placeholders with your preferred information.
20. After editing you should reload the policy using `apparmor_parser` with the `-r` flag to indicate it is a reload operation.

This task continues in the Deliverable.



Developing profiles

Your tasks are to develop appropriate scripts and AppArmor profile to satisfy the following requirements:

1. What changes do you need to make to the profile to enable the original operation of the script before we added a profile?
2. Modify the script to add a line (such as the date) to the file `script.log` every time its run. This log should be in a defined place - for example given the above `/home/cnd/testing/script.log`
3. Add the following lines
`echo " Reading Log" cat script.log echo "Logging Append" date && script.log`
4. Ensure that the `script.log` can only be written to (**without** using `chflag`)

**Useful Resources**

The following resources may be of use in your completion of this task:

- [How to create an AppArmor Profile](#)
- [Using AppArmor](#)
- [Debugging Apparmor](#)
- [AppArmor_Core_Policy_Reference](#)

**Exploring Further**

AppArmor also provides means of managing resource limits, via the `rlimit` directive. This allows you to limit the amount of diskspace, filesize, cpu or even memory that a process can use.

More details on this can be found in this [article](#).

3 MySQL Hardening

MySQL is a very popular database server. As a result there is a wealth of information available to support it. You should explore some of this material, using the provided resources as a starting point. The [MySQL Security Guide](#) is an excellent starting point.

Having consulted the security guide undertake the tasks that follow, taking note of additional resources provided.

Enable TLS

Enable the ability to turn on TLS protection for network connections to your database. The easiest way to test this is connecting from a second client using the `mysql` command at the command prompt. Alternately install the MySQL workbench software in Windows. If you use this you should still be comfortable working with MySQL at the command prompt. You should verify the connection is encrypted with Wireshark/tcpdump.



To TLS or Not to TLS, that is the question¹

Some may argue that TLS is moot and not needed. This is certain true in the case of our configuration, where the web server, web application and database are hosted on the same 'physical' server. As such there is an alternate means for connecting via `localhost` or via a socket interface. Traffic via `localhost` is however still vulnerable to snooping if the system is compromised.

In more realistic and larger scale deployments the DB and web layer are typically separate, and may not even reside in the same DMZ.

As with many things security related, *it depends* as you need to consider threats to the environment and the risk of those being realised.

Some starting resources are:

- [MySQL - Using Encrypted Connections](#)
- [Configuring MySQL to Use Secure Connections](#)
- [MySQL - encrypted connection protocols ciphers](#)
- [MySQL - secure socket layer support](#)
- [Encrypted connections with TLS protocols](#)
- [MySQL TLS performance](#)

As a **quick start**, you may find the following snippet of the `mysql` server configuration file a good starting point. Be sure to update the fields to match your configuration (and especially the location of the certificates you previously generated).

```
[mysqld]
ssl_ca = "cacert.pem"
ssl_capath = "/.../ca_directory"
ssl_cert = "server-cert.pem"
ssl_cipher = "DHE-RSA-AES256-SHA"
ssl_crl = "crl-server-revoked.crl"
ssl_crlpath = "/.../crl_directory"
ssl_key = "server-key.pem"
```

NOTE: The `ssl_capath` and `ssl_crlpath` **must** be fully specified paths.

¹With apologies to the [Bard](#) and former prince of Denmark



You may find there is an issue with your Wordpress installation communicating with the DB server once encryption is enabled. This may need to be tweaked. The following [article](#) is of use in solving this. Once you have tested the setup, and if you are having issues with Wordpress, you can **disable encryption** until it is needed.



SSL failed on Mysql?

If your Mysql fails to start, and/or fails to support the TLS/SSL configuration you have added, check the log file. You may see entries such as:

- Failed to initialize TLS for channel: mysql_main. See below for the description of exact issue.
- Failed to set up SSL because of the following SSL library error:
SSL_CTX_set_default_verify_paths failed
- Plugin mysqlx reported: 'Failed at SSL configuration: "SSL context is not usable without certificate and private key"'

The most likely cause of this is that the Mysql server user (mysql) cannot read the files. you can test this with the command:

```
sudo -u mysql cat /path/to/ssl/cert/file.pem
```

You need to replace the path in the command above with what matches your setup. If the files cannot be read, ensure you have set file (and directory) permissions correctly.



Validating Secure Communication

You can use the `ssllscan` tool to validate the mysql sever is offering secure communications. Alternately you can inspect traffic using a tool such as Wireshark. The easiest way to test is making a connection to your DB server instance from a mysql client on a **separate** machine. This ensures the traffic transits the network rather than via the loopback interface.

General Hardening

A variety of approaches have been discussed in lectures as to how to secure MySQL. Work through these recommendation using the following links as starting point.

- [Security Considerations for LOAD DATA LOCAL](#)
- [W3: MySQL Security](#)
- [5 Essential Steps to Hardening Your MySQL Database](#)
- [DevSec MySQL Baseline](#)
- [Boost the Security of a MySQL Database](#)
- [Making MySQL Secure Against Attackers](#)



Tired of Reading?

Here are two good videos you can watch on the topic

- [Hardening MySQL: MySQL Security Basics - David Busby](#) ⌚ 20:19
- [MySQL Security Best Practice](#) ⌚ 41:20

Remember you can often use chapter markers or increase the speed when watching video.

**Securing MySQL Quickstart**

Put together a set of notes for yourself as a reminder of how to go about hardening, and testing a MySQL installation that the basics have been followed. This should be more than just running the built-in script.

Extension: If you are feeling confident, how many of these could you put into a script, or series of scripts?

**Done?**

A quick checklist to confirm you have covered the required bases

1. DB server configured with a TLS certificate
2. Server Only accepting secured connections
3. Successful connection using encrypted link over a network (virtual or localhost)
4. User security considered
5. Permissions considered
6. Basic hardening undertaken

**MariaDB**

MariaDB diverges a little from Myql in terms of its TLS setup. If you are interested, the following may be of use. However this course is making use of Mysql **only**.

- [Securing Connections for Client and Server](#)
- [How to set up MariaDB SSL and secure connections from clients](#)

4 WordPress Security

WordPress is a popular content management system (CMS). While initially intended as a blogging platform, it has developed to extend this functionality being a site wide CMS. However it suffers from a number of issues, which are directly linked to its popularity, and perceived 'ease of use'. Often regarded as 'insecure'

In a typical environment every user is given WordPress administrator role, even when the user just needs to input content which was written by someone else. The same applies to files and directory permissions; all of which are configured with the less restrictive permissions.

WordPress site administrators often take such approach because it is the easy way out. Rather than checking what the user needs to do, we prefer to just assign him the administrator role so he or she can do his work without asking for any assistance. The same with files and directories permissions. For example some WordPress plugins store cache files or other data on the file system, i.e. in the WordPress directories. It is easier to simply configure 777 permissions in the `/wp-content/plugins/` directory because all plugins will work and you do not have to stay troubleshooting and tweaking permissions each time you install a new plugin.

Hence to avoid a bit of extra work and troubleshooting, and because allowing "all privileges" will always work many tend to **ignore the principle of least privilege** and as a result leave their WordPress installations open for malicious hack attacks.



Using online resources that you find, and notes from the lecture consider how you would secure your WordPress installation. You can maintain your notes (which will be of use later) under the following headings:

- Users
- Database
- Filesystem
- Webserver

While you are encouraged to try implement some of the advice you find, and verify your installation still works, this is **not** required.

Keep notes of your discussion.



There are a number of plugins specifically focused on WordPress Security. Investigate some of these and see what functionality they offer. Consider their ease of installation, and ease of use. A good place to start your exploration is [The Best WordPress Security Plugins To Lock Out Malicious Threats](#).

- Consider which of these are free to use?
- What plugin/configuration would you recommend to a small local coffee shop, where the owner maintains the site, but does not have an IT background?
- Discuss your approach with others.
- **Keep notes of your discussion.**

Extension: Try implement some of the advice you find, and verify your installation still works. Keep notes on your experience.



2FA in WordPress

As discussed in lectures, there are offerings that allow you to implement token based 2FA in WordPress. Three ones to look at and consider for implementation are:

- [Duo Two-Factor Authentication](#)
- [Google Authenticator](#)
- [Two Factor Authentication](#)

Think about the following issues relating to 2FA, and discuss with others in your class:

- How does your soft token on your phone handle multiple sites?
- How would you consider backing up your 2FA token?
- What would happen when 2FA is lost/damaged?
- How would you plan for recovery?

None of these need to be implemented but you should consider the issues above and how they could be addressed.

Keep notes of your discussion.



Some Additional Resources

- [Principle of Least Privileges and WordPress Security](#)
- [WP Activity Log Plugin](#)
- [Guide to Configure Secure Database Permissions](#)
- [How to Add an Admin User to the WordPress Database via MySQL](#)
- [Restricting User Privileges on WordPress Databases \(via archive.org\)](#)
- [Secure WordPress MySQL database with restricted privileges](#)
- [Why minimum MySQL user WordPress database privileges improve security](#)
- [How to Restrict WordPress Admin Access by IP Address](#)

5 Email with SSMTP

This task must be completed as future activities depend on it.

In a production or hosted environments you would have a system with a publicly reachable IP address, and functional forward and reverse DNS. There are some of the basic requirements for being able to run a fully functional Mail Transport Agent (MTA) - ie the ability to send email directly to endpoints.

For the purposes of this tutorial (and as something useful if you do not want to go through setting up a full MTA), we will be setting up our Ubuntu server to act as a mail client. This means it relays mail onto a dedicated mail server for onward delivery.

Having the ability to send email, allows us to start building log notification as well as the automation of submission of events into ticketing or SMS alerting systems. We will be using this for some log processing later in the course.

The package we are using is `ssmtp`. `ssmtp` is a very simple MTA to deliver mail from a computer to a mail hub (SMTP server). `ssmtp` is simple and lightweight, there are no daemons or anything hogging up CPU. `ssmtp` does not receive mail, expand aliases, or manage a queue. Installation can be done using the following:

```
$ sudo apt install ssmtp
[sudo] password for cnd:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libgnutls-openssl27
The following NEW packages will be installed:
  libgnutls-openssl27 ssmtp
0 upgraded, 2 newly installed, 0 to remove and 22 not upgraded.
Need to get 64.4 kB of archives.
After this operation, 133 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Configuration

After installation you need to configure the software. To be able to use it you need to have appropriate email server details. Most email servers (possibly not those **inside** a corporate environment) will not send email for unknown clients. A solution to this is the use of client authentication via a username and password - SMTP-AUTH. This functionality is now widely supported, and is what is leveraged by the `ssmtp` tool. Alternately if you have a mail server that does not require this, you just need to configure that as the `mailhub`.

One of the easiest ways to configure the setup is to use a free online email account (such as Gmail) to provide authentication and relay sent mails to their destination.



Finding Host Settings

You will need to find the appropriate server name and port numbers to connect to. Microsoft provides a nice overview of the settings for [popular online providers](#). You are looking for the SMTP server or SMTP submission server. These typically run on ports 465/tcp (SSL/TLS) or 587/tcp (STARTTLS) depending on what encryption mechanism they are using.

Microsoft Office 365

Server: `smtp.office365.com`

Port: `587`

Encryption: `STARTTLS`

Google Gmail

Server: `smtp.gmail.com`

Port: `465` Encryption: `SSL/TLS`

Port: `587` Encryption: `STARTTLS`

Make sure you know your username and password before continuing.

The ssmtp is located at `/etc/ssmtp/ssmtp.conf`, and needs to be edited to contain the appropriate username, password and server settings. A sample config can be found on the next page. You will need to use `sudo` to edit the configuration files.



Important Notice

- If you are using Gmail you will need to enable App passwords to be able to authenticate, as this is done using a username + password rather than Google's preferred approach of OAuth. Your account **must** have 2FA enabled for this to be available.
- **This App Password must be entered without spaces!**
- You can set up a secondary 'disposable' Gmail account to use.
- The Noroff provided 0365 email addresses are also unlikely to work given the enforced 2FA.
- Follow the instructions that **your** chosen webmail provider provides for authenticating.

An example of a complete configuration file for ssmtp is shown below, configured to use Gmail.

```
# Config file for sSMTP sendmail
#
# The person who gets all mail for userids < 1000
# Make this empty to disable rewriting.
#root=postmaster
root=MyEmailAddress@gmail.com

# The place where the mail goes. The actual machine name is required no
# MX records are consulted. Commonly mailhosts are named mail.domain.com
#mailhub=mail
mailhub=smtp.gmail.com:587

AuthUser=MyEmailAddress@gmail.com
# App password generated - set it up for Mail on Google
AuthPass=dwysmagukuoiduve #change this!

# Where will the mail seem to come from?
# make sure this matched your setup.
rewriteDomain=gmail.com

# The full hostname
#hostname=MyMediaServer.home
# A FQDN hostname means you ahve workign DNS
# set this to localhost if you have issues esp with Google.
hostname=your-fqdn-hostname

# Are users allowed to set their own From: address?
# YES - Allow the user to specify their own From: address
# NO - Use the system generated From: address
# Be aware of the security risk for this on multi-user systems.
FromLineOverride=YES
```

Sending Email

Once you've configured `ssmtp` it's time to try and send an email. The simplest way to do this is to run `ssmtp` in a terminal with a recipient email address. So:

```
ssmtp recipient_email@example.com
```

`ssmtp` will then wait for you to type your message, which needs to be formatted like this:

```
To: recipient_email@example.com
From: myemailaddress@gmail.com
Subject: test email
```

```
Hello World!
```

Note the **blank line** after the subject field. Everything you type from the `Hello World!` onward is the body of the email. Once you have finished composing your email hit `Ctrl-D` (indicating end of file). After a few seconds `ssmtp` will send the message, and you should be able to verify its received. If you pre-prepare a file to be passed as input to `ssmtp` the end-of-file indicates the completion of the message.

You should check for errors and other details in the log files `/var/log/mail.log` and `/var/log/mail.err`. A successful send using Gmail looks similar to the entry below:

```
Sep 17 16:57:31 cnd22 sSMTP[30150]: Creating SSL connection to host
Sep 17 16:57:31 cnd22 sSMTP[30150]: SSL connection using ECDHE_RSA_AES_256_GCM_SHA384
Sep 17 16:57:36 cnd22 sSMTP[30150]: Sent mail for root@localhost (221 2.0.0 closing connection
i12-20020ac2522c000000b004a03e7e8019sm1181651fl.289 - gsmtplib) uid=0 username=root outbytes=382
```

The usual case for Gmail authentication errors looks like:

```
Sep 17 16:56:38 cnd22 sSMTP[30138]: Creating SSL connection to host
Sep 17 16:56:38 cnd22 sSMTP[30138]: SSL connection using ECDHE_RSA_AES_256_GCM_SHA384
Sep 17 16:56:38 cnd22 sSMTP[30138]: Authorization failed (535 5.7.8
https://support.google.com/mail/?p=BadCredentials
u10020ac258ca000000b00498ebd60c35sm1015233lfo.165 - gsmtplib)
```



Log monitoring

you can use the command `tail -f logfile` to watch a log live that will scroll as new entries are added. Its useful to run this in a second SSH session or terminal so you can watch what happens as you run commands, without having to keep going back and checking what is in the logs.

This is however not much use for non interactive sending. Fortunately we can use the power of Unix file redirection to solve the problem. You can pre-generate correctly formatted messages, or have a script generate them and store in a file. This can then be used along with the following syntax to send the message. A command line pipe (`|`) can also be used depending on how you approach this.

```
/usr/sbin/ssmtp myemail@myaddress.com </home/cnd/MyScripts/msg.txt
```

Once you have configured `sSMTP` it becomes the default "email client" and so you'll start receiving relevant output from any Cron jobs that execute on your system too. To ensure this works smoothly you will most likely need to install the `mailutils` package. After doing this you can use syntax similar to that below for sending messages, which may be more convenient.

```
$ echo "Program executed ok!" | mail -s "Testing" random.user@example.org
```

If your *subject* (the entry after `-s`) has a space in it remember to enclose it in quotes.

**Check your work**

You should ensure you have completed the following:

- Can send email manually to an external email account
- Can send email using a pipe or redirect from the commandline

**No delivery?**

If mail is not appearing in your inbox, make sure you check your junk folder on the email account you are sending to. This is especially if the logs indicate that delivery was successful.

**Adding reverse aliases**

A reverse alias changes the "From" address. This means you can make the email appear as if it's from a different email address. Some additional configuration may be needed on your mail provider to enable this as it means that the From address differs from the address being used for authenticating. If you need to change this or wish to experiment, the configuration can be set up for `ssmtp` in the file `/etc/ssmtp/revaliases`, and you should add something similar to the line following. Consult the `ssmtp` documentation for more details.

```
root:YourFromName@gmail.com:smtp.gmail.com:587
```

**Advanced Configuration options**

An alternative to `ssmtp`, you can install a more full features MTA such as Postfix and configure it to also serve as a email stub (send only) node. This has a number of advantages including the powerful functionality built in such as header manipulation, filtering etc.

Two good starting points for this are:

- [How To Install and Configure Postfix as a Send-Only SMTP Server on Ubuntu 20.04](#)
- [Configure Postfix to Use Gmail SMTP on Ubuntu 20.04](#)



Be aware that setting up a full featured MTA is quite an involved process and requires a number of other external setups if you do not have directly exposed port 25/tcp. Setting this up can also expose a system to risk of being exploited as an **open relay**. DNS configuration is also typically required in order for a system to send and receive email.

6 Benchmarking Crypto

Cryptographic algorithms have varying performance. The purpose of this task is to gain an understanding and appreciation of the impact that the use of different cryptographic algorithms have on the processing speed. A particular focus is to gain an understating and appreciation of the advantages offered by hardware acceleration of cryptographic operations. the most common of these is the Advanced Encryption Standard, better known as AES. Various vendors offer other hardware accelerator cards for a range of algorithms. these are particularly common in high volume web servers where much of the RSA processing is handled in discrete ad-din hardware, rather than on the system CPU.

This task should not be long or onerous, and can be completed with little effort .

Keep notes of your commands and the results you obtain.

AES benchmarking

Moderns CPU systems support hardware accelerated AES encryption, first introduced in early 2008, this is often referred to as the **AES-NI** instruction on Intel Processors. This is supported on a number of other CPU variants. This can be harnessed by software such as the Linux Kernel and other libraries such as openssl. The ability to have native hardware supported encryption has removed what was previously a major argument against using strong encryption for protecting data both at rest, and in transit.

In Linux you can verify if your CPU is reporting support for the feature by looking at the flags field in `/proc/cpuinfo`. The following command will see if the function is reported on your system:

```
cat /proc/cpuinfo | grep -o aes
```

This will return `aes` which indicates if the CPU supports the AES-NI flag, showing hardware acceleration is present, and can be used.

How fast ?

The `openssl speed` command can be used to benchmark performance on your system for various cryptographic algorithms. An example is shown below for the `aes-128-gcm`² cipher suite.

```
cnd@cnd2022:~$ openssl speed -elapsed -evp aes-128-gcm
You have chosen to measure elapsed time instead of user CPU time.
Doing aes-128-gcm for 3s on 16 size blocks: 39823318 aes-128-gcm's in 3.00s
Doing aes-128-gcm for 3s on 64 size blocks: 25069552 aes-128-gcm's in 3.00s
Doing aes-128-gcm for 3s on 256 size blocks: 9335022 aes-128-gcm's in 3.00s
Doing aes-128-gcm for 3s on 1024 size blocks: 2771629 aes-128-gcm's in 3.00s
Doing aes-128-gcm for 3s on 8192 size blocks: 342000 aes-128-gcm's in 3.01s
Doing aes-128-gcm for 3s on 16384 size blocks: 104507 aes-128-gcm's in 3.02s
OpenSSL 1.1.1f 31 Mar 2020
built on: Mon Aug 23 17:02:39 2021 UTC
options:bn(64,64) rc4(16x,int) des(int) aes(partial) blowfish(ptr)
The 'numbers' are in 1000s of bytes per second processed.
type                16 bytes      64 bytes      256 bytes    1024 bytes    8192 bytes   16384 bytes
aes-128-gcm         212391.03k   534817.11k   796588.54k   946049.37k   930785.38k  566967.78k
```

²https://en.wikipedia.org/wiki/Galois/Counter_Mode



Other Ciphers

Run the tests above for at least the following ciphers and log your results.

- AES CBC - aes-128-cbc aes-192-cbc aes-256-cbc
- AES ECB - aes-128-ecb aes-192-ecb aes-256-ecb
- AES GCM - aes-128-gcm aes-192-gcm aes-256-gcm

Build a table up to compare the results and answer the following:

1. How fast are the algorithms relative to each other?
HINT: The easiest way to compare is to take the fastest, and use it to divide all the other results by. This means the fastest will have a score of 1.0 and the others will be relative to that. This is called normalisation, and is a way of making data a lot more understandable. Having your data in a spreadsheet makes this very easy to do.
2. Which of ECB, CBC and GCM is the fastest for each key size, where the keys are 128, 192 and 256 bit?
3. Draw a graph of your results. You should use any tool you find appropriate. If you are unsure, Excel is a great place to start.

Keep notes of your commands and the results you obtain.

Practical Speed Impact

This exercise evaluates the practical use of encryption and differing algorithms in undertaking the encryption. To start with you will need an appropriate input file. Make sure you run this on a file-system that has at least 2x the drive space available. *You can delete these files when the tasks are complete.* The command below will generate a 1GB file you can use as a basis for the remainder of this activity. You should ideally not use a file size smaller than 500MB.

```
$ dd if=/dev/zero bs=1M count=1024 of=1g.bin
```

Output should look similar to that below:

```
$ time dd if=/dev/zero bs=1M count=1024 of=1g.bin
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 1.33248 s, 806 MB/s

real    0m1.469s
user    0m0.004s
sys     0m1.439s
```

The resulting file should look something like:

```
$ ls -la 1g.bin
-rw-rw-r-- 1 cnd cnd 1073741824 Sep 12 20:26 1g.bin
```

Running Tests

You can use a command similar to the following can be used to run timing tests on the various algorithms. This tells openssl what algorithm to use, what the input and output files are, and to automatically use a key of *pass* for the encryption (but to perturbate it using the *pbkdf2* scheme to suppress warnings).

```
time openssl enc -aes-256-cbc -in 1g.bin -out 1g.bin -k pass -pbkdf2
```

The value you are interested in here is the *real* amount of time elapsed. You are **strongly** encouraged to run this a couple of times to get reasonable and stable results, ideally discard the fastest, and slowest and average the remainder.

**My results dont match?**

The specific results you get will differ depending on the exact hardware you are running on (as well as other load related factors - especially in virtual environments). The two biggest contributors are your CPU and what backend storage you are using. A NVMe or SSD will be significantly faster than a hard disk. Similarly a faster (and more modern CPU) will run faster. Your results should however be relative to each other approximately the same as other people are getting.

**The need for speed...**

Using at least five (5) **different** algorithm families supported by `openssl`, benchmark how they perform against each other encrypting the same input file. You should try and vary the key sizes within these. You can find what is supported by running `openssl help` and `openssl help enc`.

Answer the following:

1. Which is the fastest algorithm overall?
2. Which is the slowest algorithm overall ?
3. How much of an impact do differing key sizes make to your algorithms?
4. How does this compare to your initial observations with AES?

Keep notes of your commands and the results you obtain.

Hashing Speed

Encryption is only one part of what makes up modern cryptography. Equally important, and in some cases more so is cryptographic hashing (often just referred to as hashing). This is important for being able to validate the integrity of files, and is something relied upon heavily when we start looking at system auditing.

You can use a command similar to the following can be used to run timing tests on the various hashing algorithms. This tells `openssl` what algorithm to use, what the input files are.

```
time openssl dgst -md5 1g.bin
```

The value you are interested in here is the *real* amount of time elapsed. You are **strongly** encouraged to run this a couple of times to get reasonable and stable results.

**The need for speed...**

Using the MD5, SHA1, SHA256, SHA3-256 and BLAKE2B512 algorithms supported by `openssl`, benchmark how they perform against each other encrypting the same input file. You can find what is supported by running `openssl help` and `openssl help enc`.

Answer the following:

1. Which is the fastest hashing algorithm, regardless of keysize?
2. Rank the algorithms in speed order, fastest first.
3. How much of an impact do differing hash output sizes make?

Keep notes of your commands and the results you obtain.



Extension exercises

The following two activities are offered as extension exercises only.

Does AES-NI make a difference?

If your CPU supports AES-NI you should be seeing reasonably fast results. Its useful to quantify what impact it makes when disabling this hardware acceleration. You can do this in software by using the following command:

```
OPENSSL_ia32cap=" 0x2000002000000000" openssl speed -elapsed -evp aes-128-gcm
```

The OPENSSL_ia32cap prefix allows you to set a mask over what is reported by the CPU hardware. Running this then **disables** the AES acceleration for the command following:

```
$ OPENSSL_ia32cap=~0x2000002000000000" openssl speed -elapsed -evp aes-128-gcm
```



Slow Down

1. Extend the results table you previously produced by running the algorithms again with the AES acceleration disabled. An example of the (unnormalised) results are shown in the table below.
2. Replot your graph
3. Approximately how much slower on average are the ciphers with AES disabled?
4. For each size, what is the average speedup you observe when using the hardware acceleration? Calculate this as (HW accelerated/software only)
5. Which of the three modes of AES you have evaluated is impacted most?
6. What advantages do you think we are able to accrue from the increased performance with AES, and why is it important that this algorithm is implemented in moderns CPUS?
7. What kinds of activities would commonly benefit from the kind of performance gains we see?

AES	type	16 bytes	64 bytes	256 bytes	1024 bytes	8192 bytes	16384 bytes
HW	aes-128-gcm	212391.03k	534817.11k	796588.54k	946049.37k	930785.38k	566967.78k
dis	aes-128-gcm	74222.57k	86034.22k	101126.55k	108445.70k	111899.99k	110193.32k

HASH acceleration

Recent Generation 10/11/12/13 Intel, AMD Zen architecture and Apple M1/M2 CPUs support the newer [SHA hashing extensions](#) which has support for the new SHA hashes in hardware similar to the AES-NI providing a significant speedup. You can determine what model CPU you have by running the command:

```
$ cat /proc/cpuinfo | grep "model name"
model name : Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz
```

In this case it is an Intel i9-9900K which means its an i9 CPU, but is a 9th generation (based on the 9900).

NOTE: The number of lines returned above will depend on how many CPU cores you have configured your VM server to have. The output will also reflect the CPU **YOU** have.

- Does your CPU support them?
- How would you establish if the are supported?
- Can you find a way of testing how much of a performance gain this gives?

Intel's [whitepaper](#) on the extensions contains more details and benchmarks.

7 TLS Tuning

SSL versus TLS

Web security has had a mixed history, in recent years there have been a number of attacks against the older SSL and TLS protocols. As standards evolve, one needs to make sure servers are correctly configured. SSL is old and insecure. Therefore version 2 and 3 should be disabled. TLS is fine, with TLSv1.1 and TLSv1.2 being preferred. TLSv1 might be still needed, for older browsers or some tooling (e.g. some older versions of wget), especially where there is legacy equipment such as embedded systems that need to be supported).

Crypto Compatibility

The Mozilla foundation maintains a [TLS compatibility](#) list of supported and recommended cipher suites. Cipher suites are a combination of the various modes of operation and combinations of hashing and encryption algorithms. The two most useful of these are:

1. [Modern Compatibility](#)
For services with clients that support TLS 1.3 and do not need backward compatibility. Provides an extremely high level of security.
2. [Intermediate Compatibility](#)
For services that don't need compatibility with legacy clients, such as Windows XP or old versions of OpenSSL. This is the recommended configuration for the vast majority of services, as it is highly secure and compatible with nearly every client released in the last five (or more) years.

Apache Suites

The Apache webserver contains its SSL/TLS configuration options in the file `/etc/apache2/mods-available/ssl.conf` which is where you will need to make changes.



SSL Support Enabled?

For any of the following to work, you need to ensure that SSL has been enabled in Apache. You can confirm that the `mod_security` has been installed and enabled using the following:

- Check Modules are installed

```
$ sudo apt-get install libapache2-mod-security2
```
- Enable additional modules

```
$ sudo a2enmod rewrite ssl security2
```
- Restart Apache 2

```
$ sudo systemctl restart apache2
```

The primary parameters that need tweaking to change systems (along with the Ubuntu defaults) are:

- `SSLCipherSuite`
- `SSLHonorCipherOrder`
- `SSLProtocol`

**Over to you**

Implement the recommended settings for the following scenarios, keeping careful note of the relevant configuration lines.

1. Mozilla Modern
2. Mozilla Intermediate
3. TLS 1.0 only
4. SSLv3 only

You should provide suitable means of testing your configuration (bearing in mind you do not have access to sites such as SSLlabs to score your configuration). Make sure you know how to test for support for a specified protocol from the command line.

You can use the `curl` utility to do your testing, which results in output similar to that below. **Refer to the `curl` man page for details.** `curl` offers a range of options to allow you limit the security /protocol level when talking/negotiating with the remote server.

```
* Trying 192.168.1.2:443...
* TCP_NODELAY set
* Connected to www.example.com (192.168.1.2) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/certs/ca-certificates.crt
*   CPath: /etc/ssl/certs
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES128-GCM-SHA256
* ALPN, server did not agree to a protocol
* Server certificate:
*   subject: C=NO; L=KRISTIANSAND; O=CND AS; CN=*.example.com
*   start date: Sep 8 00:00:00 2023 GMT
*   expire date: Oct 26 12:00:00 2024 GMT
*   subjectAltName: host "www.example.com" matched cert's "*.example.com"
*   issuer: C=US; O=DigiCert Inc; CN=DigiCert SHA2 Secure Server CA
*   SSL certificate verify ok.
> HEAD / HTTP/1.1
> Host: www.example.com
> User-Agent: curl/7.68.0
> Accept: */*
```



Other tools to test with

Other tools you can use for testing ciphers are `openssl s_client` and `ssllscan` which you can install using `apt`.

An *example* of output produced by `ssllscan` is:

Testing SSL server `www.noroff.no` on port 443 using SNI name `www.noroff.no`

SSL/TLS Protocols:

```
SSLv2      disabled
SSLv3      disabled
TLSv1.0    disabled
TLSv1.1    disabled
TLSv1.2    enabled
TLSv1.3    enabled
```

TLS Fallback SCSV:

Server supports TLS Fallback SCSV

TLS renegotiation:

Session renegotiation not supported

TLS Compression:

OpenSSL version does not support compression
Rebuild with `zlib1g-dev` package for zlib support

Heartbleed:

TLSv1.3 not vulnerable to heartbleed
TLSv1.2 not vulnerable to heartbleed

Supported Server Cipher(s):

```
Preferred TLSv1.3 128 bits TLS_AES_128_GCM_SHA256      Curve 25519 DHE 253
Accepted  TLSv1.3 256 bits TLS_AES_256_GCM_SHA384      Curve 25519 DHE 253
Accepted  TLSv1.3 256 bits TLS_CHACHA20_POLY1305_SHA256 Curve 25519 DHE 253
Preferred TLSv1.2 128 bits ECDHE-RSA-AES128-GCM-SHA256 Curve 25519 DHE 253
Accepted  TLSv1.2 256 bits ECDHE-RSA-AES256-GCM-SHA384 Curve 25519 DHE 253
Accepted  TLSv1.2 256 bits ECDHE-RSA-CHACHA20-POLY1305 Curve 25519 DHE 253
Accepted  TLSv1.2 128 bits AES128-GCM-SHA256
Accepted  TLSv1.2 256 bits AES256-GCM-SHA384
```

Server Key Exchange Group(s):

```
TLSv1.3 128 bits secp256r1 (NIST P-256)
TLSv1.3 128 bits x25519
TLSv1.2 128 bits secp256r1 (NIST P-256)
TLSv1.2 128 bits x25519
```

SSL Certificate:

Signature Algorithm: `sha256WithRSAEncryption`
RSA Key Strength: 2048

Subject: `*.noroff.no`

Altnames: `DNS:*.noroff.no, DNS:noroff.no`

Issuer: `DigiCert TLS RSA SHA256 2020 CA1`

Not valid before: `Sep 27 00:00:00 2021 GMT`

Not valid after: `Oct 27 23:59:59 2022 GMT`



What TLS cipher suite(s) do the following sites support?

- Noroff's Webservers - is there a difference between `www.noroff.no` and `noroff.no` ?
- Moodle
- Komplett
- vg.no
- Norwegian Parliament

What other sites can you check? Have you found any surprised (either of a very high security level, or still supporting SSL?)

See the extension to these questions in the **Deliverable** for the week.



TLS 1.3 support

The minimum version of the Apache Web Server must be - Apache 2.4.37 and OpenSSL version must be OpenSSL 1.1.1. you can check your versions by using the commands `apache2 -v` and `openssl version`

This should not be a problem on the server we have been configuring, but is worth keeping in mind if working on other systems.



Client Security

You may want to conduct a test of your web client security. Qualys provides a [service](#) which does something similar to `ssllscan` but in reverse, evaluating the capabilities of the web client (typically your browser).



Resources

- [Disabling TLS 1.0 on Apache](#)
- [Apache Documentation - Ciphersuites](#)
- [Optimize SSL/TLS for Maximum Security and Speed](#)
- [The TLS handshake explained](#) ⌚ 16:58

8 Deliverable



These deliverables are **in scope** for and expected to be **complete** and available prior to commencing Engagement Quiz 2.

As previously you are strongly encouraged to make sure you keep appropriate notes and make sure these are accessible when you take the engagement Quiz.

Theoretical Aspects

The follow are checks for you to ensure you understand theoretical aspects covered in this tutorial, along with having completed the supporting practical activities.

Certificate generation

Make sure you understand the operation of TLS certificates and are able to answer the following:

1. What would the **advantage** be of using a certificate from a known CA rather than the self-signed one we have used in the tutorial.
2. What does a certificate attest to about a server?
3. On what basis do we know to trust a server?
4. How could a server trust a client ?

Discuss your answers with others in your class.

Security Headers

Ensure that you clearly understand the purpose, function, and appropriate configuration of the headers discussed in lectures and in this tutorial. Make appropriate notes for yourself. You should have completed the activities noted in the Task.

Wordpress - 2FA in considerations

Think about the following issues relating to 2FA, and discuss with others in your class:

- How does your soft token on your phone handle multiple sites?
- How would you consider backing up your 2FA token?
- What would happen when 2FA is lost/damaged? How would you plan for recovery?

Keep notes of your discussion.

Practical Deliverable

You have two practical tasks to complete for this week's deliverable. the first is a simple bash shell script, and the second is the collection of some data

TLS Ciphers

Develop a short and **simple** shell script using `bash` to automate query and processing of the protocol checks for a host/website. The script should take a domain name/website as a **parameter** on the command line. It should be able to handle inputs such as the examples below:

- `$ tlscheck https://noroff.no`

- \$ `tlscheck http://noroff.no`
- \$ `tlscheck noroff.no`

This should work for hosts both on the internet and on a local isolated network. Clearly state as a comment in the script any dependencies. you must use **only** tools/methods covered in the course thus far.

*HINT: This is quite feasible using only the skills covered in the first two weeks relating to text manipulation along with content covered this week. Do not overthink the problem! Get it working on the command line **first** and build up your script.*

Output **must** match the following (replacing xxxxxx with **your** email address) for all cases above:

```
tlscheck for noroff.no
Written by xxxxxx@stud.noroff.no
SSLv2      disabled
SSLv3      disabled
TLSv1.0    disabled
TLSv1.1    disabled
TLSv1.2    enabled
TLSv1.3    enabled
```

Once you have the basic script in place, you need to implement the following:

1. Check if an `http://` hostname/URL is provided, and if so exit with an appropriate error message
2. If a hostname is provide without a protocol, a warning should be issued and `https://` assumed.
3. Adjust your script to make it run faster as well as reduce/remove any unnecessary tests that may be conducted? Add appropriate comments to this effect to your script.



Check your work

Check your work by comparing the results from your script with the analysis you did earlier in the course relating to what TLS cipher suite(s) do the following sites support?

- Noroff's Webservers - is there a difference between `www.noroff.no` and `noroff.no` ?
- Moodle
- Komplett
- vg.no
- Norwegian Parliament

Crypto

You should have the material and notes you prepared undertaking the Crypto task. In summary you should have the following:

1. Scores of various algorithms
2. Graph of crypto benchmarks
3. Ranking in terms of speed.
4. Scores for hashing algorithms
5. Graph of hashing benchmarks. This **must** contain your `@stud.noroff.no` address and name on the image.
6. Ranking in terms of speeds

Your final deliverable task is to ensure you are fully caught up with the first four weeks tutorials. Future tutorials will be focusing more on problem solving than just the configuring of systems.

9 Preparation

As noted previously there seems to be a general weakness when dealing with networking concepts. These however form the basis of much of the work to be covered in the second part of the semester.

You must ensure that you are comfortable with the following networking concepts, as they are **assumed knowledge**, having been covered in your previous courses. **At a minimum, you need to be comfortable with the following:**

1. IPv4 Addressing
2. IPv4 Subnets and the process of subnetting (splitting) and supernetting (merging) of network blocks.
3. Understand CIDR and Dotted Quad netmasks
4. Be able to determine if two IPv4 addresses are in the same network given a netblock, or addresses and a netmask.
5. Determine if an IP address falls within a given Netblock.
6. How traffic flows on a IPv4 Network (hop by hop)
7. The basic of Traffic transmission on an Ethernet network, including ARP, basic routing and local subnet validation
8. Firewall operation concepts - access controls, routing and architecture
9. Principles around basic firewall rules, including direction of flow, network interfaces and what kind of elements within an IPv4 datagram that can be used to 'select' and filter datagrams arriving at the firewall.
10. Be able to express at a *high level* a firewall rule given a description of traffic. For example, what rules would you need if we needed to allow traffic from clients on the internet to a web server?
11. The principles and concepts relating to Virtual Private Networks.
12. What the benefits of these are, along with how and where they should be used.
13. Understand how network packets are constructed, and how they appear when traversing a network.
14. Be able to load, capture and inspect packets in Wireshark.
15. Refresh your view on basic concepts relating to networking and network security

These concepts have all been covered in prior courses, and you should review your notes and course material.

A reminder that refresher material was made available at the start of the course. You are encouraged to use this.

The concepts above will be relied on as we transition to the more network specific elements of Computer Network Defence.

A News

The following are new articles of interest in the last week. These are all relatively short, and you should get in the habit of staying aware of what is going on in the industry at large. The topics discussed here fall in scope for the engagement tasks.



News Week of 18 September 2023

APT [Password spray campaigns enable intelligence collection at high-value targets](#)
Since February 2023, Microsoft has observed password spray activity against thousands of organisations carried out by an actor we track as Peach Sandstorm (HOLMIUM). Peach Sandstorm is an Iranian nation-state threat actor who has recently pursued organisations in the satellite, defence, and pharmaceutical sectors around the globe. Based upon the profile of victim organisations targeted and the observed follow-on intrusion activity, Microsoft assesses that this initial access campaign is likely used to facilitate intelligence collection in support of Iranian state interests

MALWARE [The iPhone of a Russian journalist was infected with spyware](#)
The iPhone of the Russian journalist Galina Timchenko was compromised with NSO Group's Pegasus spyware. A joint investigation conducted by Access Now and the Citizen Lab revealed that the journalist, who is at odds with the Russian government, was infected with the surveillance software.

BREACH [Airbus Launches Investigation After Hacker Leaks Data](#)
Cybercrime intelligence firm Hudson Rock reported on Tuesday that a hacker claimed earlier this month on a cybercrime forum that they had hacked Airbus. The hacker, who recently announced joining an emerging ransomware group, apparently obtained the personal information of 3,200 people associated with Airbus.

ATTACK [Caesars reportedly paid millions to stop hackers releasing its data](#)
Caesars Entertainment reportedly paid "tens of millions of dollars" to hackers who threatened to release company data, Bloomberg has reported. The attack was reportedly perpetrated by a group called Scattered Spider (aka UNC 3944), a group skilled at using social engineering to bypass corporate network security. It's the second notable attack of a Las Vegas casino group, following a hack that caused a cyber outage at MGM Resorts

TIPS [5 Password Cracking Techniques Used in Cyber Attacks](#)
Successful data breaches can frequently be traced back to weak or stolen passwords. Research for the 2023 State of the Phish report from Proofpoint found that only a notable proportion of working adults manually enter a unique password for each work account. Worse, many of them even gave out their passwords in threat situations.

B Tools

These are some interesting tools that have popped up in the last week. These are all a use at your own risk, and are **not** part of the course but are provided as a means of further broadening your views on cyber security and cyber operations in general.



hurl

Hurl is a command line tool that runs HTTP requests defined in a simple plain text format. It can perform requests, capture values and evaluate queries on headers and body response. Hurl is very versatile: it can be used for both fetching data and testing HTTP sessions.

<https://github.com/Orange-OpenSource/hurl>



q - Text as Data

q's purpose is to bring SQL expressive power to the Linux command line and to provide easy access to text as actual data. q allows the following:

- Performing SQL-like statements directly on tabular text data, auto-caching the data in order to accelerate additional querying on the same file.
- Performing SQL statements directly on multi-file sqlite3 databases, without having to merge them or load them into memory

<https://github.com/harelba/q>



q - Text as Data

q's purpose is to bring SQL expressive power to the Linux command line and to provide easy access to text as actual data. q allows the following:

- Performing SQL-like statements directly on tabular text data, auto-caching the data in order to accelerate additional querying on the same file.
- Performing SQL statements directly on multi-file sqlite3 databases, without having to merge them or load them into memory

<https://github.com/harelba/q>



fish - the friendly interactive shell

fish is a smart and user-friendly command line shell for macOS, Linux, and the rest of the family. fish includes features like syntax highlighting, autosuggest-as-you-type, and fancy tab completions that just work, with no configuration required.

<https://github.com/fish-shell/fish-shell>

C OpenSSL speed Benchmarks Reference

This the the output of openssl speed on an Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz. This CPU is nearly 10 years old and your results should be substantially fasster on a more modern CPU.

type	16 bytes	64 bytes	256 bytes	1024 bytes	8192 bytes	16384 bytes
md2	0.00	0.00	0.00	0.00	0.00	0.00
mdc2	0.00	0.00	0.00	0.00	0.00	0.00
md4	57984.62k	169167.80k	400650.90k	673039.24k	811401.22k	808017.10k
md5	87625.29k	191380.81k	281684.33k	444269.52k	487044.64k	481815.21k
hmac(md5)	37074.50k	108814.56k	259329.88k	405720.44k	482786.45k	395341.49k
sha1	87348.26k	201164.33k	367410.34k	490307.93k	538148.42k	534206.37k
rmd160	30346.96k	74455.11k	137823.81k	177836.26k	193407.66k	191950.34k
rc4	347008.58k	501883.33k	573017.51k	602104.10k	605492.91k	600366.75k
des cbc	50662.18k	53275.47k	50159.48k	50918.49k	46806.73k	49954.71k
des ede3	17198.68k	19043.21k	18430.46k	16366.07k	19376.81k	18339.16k
idea cbc	0.00	0.00	0.00	0.00	0.00	0.00
seed cbc	60420.39k	60945.15k	54076.87k	58653.55k	61908.51k	61152.32k
rc2 cbc	33347.08k	32839.90k	33361.66k	33811.86k	34420.10k	34356.92k
rc5-32/12 cbc	0.00	0.00	0.00	0.00	0.00	0.00
blowfish cbc	85903.65k	89510.94k	87328.45k	89954.46k	90524.33k	92238.08k
cast cbc	79084.97k	85830.95k	81440.43k	66637.06k	74818.44k	82528.24k
aes-128 cbc	145172.70k	144330.55k	154090.42k	144372.39k	149182.07k	153116.43k
aes-192 cbc	110455.18k	116205.90k	115620.72k	127530.95k	136388.61k	135384.44k
aes-256 cbc	100578.23k	112626.66k	98724.34k	103487.49k	98060.97k	86363.95k
camellia-128 cbc	63136.00k	100700.03k	115182.19k	123277.65k	124296.48k	109248.51k
camellia-192 cbc	55461.67k	81094.27k	75312.71k	80402.62k	86322.86k	95054.69k
camellia-256 cbc	58449.69k	70438.25k	82064.49k	84333.42k	97197.12k	98001.61k
sha256	48057.64k	106247.30k	186729.23k	222365.88k	253678.02k	243924.33k
sha512	34558.91k	138363.38k	219134.70k	304942.41k	335902.14k	349916.21k
whirlpool	20828.87k	44894.67k	74102.17k	89242.11k	95857.32k	96408.06k
aes-128 ige	143127.05k	151613.24k	151824.81k	148740.10k	161042.66k	158628.90k
aes-192 ige	117713.46k	127186.66k	131287.21k	133632.34k	115853.99k	121740.24k
aes-256 ige	81843.01k	95262.86k	108232.52k	114725.33k	117824.60k	122082.65k
ghash	619382.28k	1045244.21k	1100646.31k	1320502.48k	1331470.34k	1221186.90k
rand	5403.43k	21037.30k	68513.42k	156215.85k	235990.90k	235818.90k
	sign	verify	sign/s	verify/s		
rsa 512 bits	0.000082s	0.000005s	12194.4	198170.9		
rsa 1024 bits	0.000231s	0.000014s	4331.5	70429.8		
rsa 2048 bits	0.001504s	0.000047s	664.9	21433.3		
rsa 3072 bits	0.004705s	0.000119s	212.6	8434.6		
rsa 4096 bits	0.011006s	0.000166s	90.9	6010.3		
rsa 7680 bits	0.091560s	0.000578s	10.9	1730.6		
rsa 15360 bits	0.498500s	0.002439s	2.0	410.0		
	sign	verify	sign/s	verify/s		
dsa 512 bits	0.000118s	0.000074s	8453.7	13444.5		
dsa 1024 bits	0.000219s	0.000186s	4560.7	5390.7		
dsa 2048 bits	0.000678s	0.000555s	1475.8	1801.6		
	sign	verify	sign/s	verify/s		
160 bits ecdsa (secp160r1)	0.0004s	0.0003s	2827.0	3139.0		
192 bits ecdsa (nistp192)	0.0004s	0.0004s	2297.4	2269.5		
224 bits ecdsa (nistp224)	0.0001s	0.0002s	10658.7	5306.2		
256 bits ecdsa (nistp256)	0.0001s	0.0002s	18779.4	5925.8		
384 bits ecdsa (nistp384)	0.0017s	0.0014s	584.0	697.0		
521 bits ecdsa (nistp521)	0.0006s	0.0011s	1779.7	901.9		
163 bits ecdsa (nistk163)	0.0005s	0.0010s	1935.8	1037.2		
233 bits ecdsa (nistk233)	0.0006s	0.0011s	1690.3	883.7		
283 bits ecdsa (nistk283)	0.0011s	0.0020s	904.4	497.4		
409 bits ecdsa (nistk409)	0.0019s	0.0036s	540.0	275.7		
571 bits ecdsa (nistk571)	0.0039s	0.0078s	255.4	127.9		

163 bits	ecdsa (nistb163)	0.0005s	0.0009s	2215.2	1168.4
233 bits	ecdsa (nistb233)	0.0006s	0.0012s	1592.3	850.1
283 bits	ecdsa (nistb283)	0.0013s	0.0023s	769.7	444.3
409 bits	ecdsa (nistb409)	0.0022s	0.0039s	444.5	253.5
571 bits	ecdsa (nistb571)	0.0045s	0.0085s	221.7	117.0
256 bits	ecdsa (brainpoolP256r1)	0.0007s	0.0007s	1350.1	1412.5
256 bits	ecdsa (brainpoolP256t1)	0.0007s	0.0006s	1495.5	1630.7
384 bits	ecdsa (brainpoolP384r1)	0.0018s	0.0013s	543.3	744.3
384 bits	ecdsa (brainpoolP384t1)	0.0017s	0.0013s	592.8	779.5
512 bits	ecdsa (brainpoolP512r1)	0.0031s	0.0025s	321.1	392.3
512 bits	ecdsa (brainpoolP512t1)	0.0030s	0.0021s	329.9	469.3
		op	op/s		
160 bits	ecdh (secp160r1)	0.0003s	3029.2		
192 bits	ecdh (nistp192)	0.0004s	2344.8		
224 bits	ecdh (nistp224)	0.0002s	6442.8		
256 bits	ecdh (nistp256)	0.0001s	7357.0		
384 bits	ecdh (nistp384)	0.0017s	602.8		
521 bits	ecdh (nistp521)	0.0007s	1362.7		
163 bits	ecdh (nistk163)	0.0004s	2242.9		
233 bits	ecdh (nistk233)	0.0005s	1893.6		
283 bits	ecdh (nistk283)	0.0010s	1008.2		
409 bits	ecdh (nistk409)	0.0019s	524.8		
571 bits	ecdh (nistk571)	0.0036s	279.4		
163 bits	ecdh (nistb163)	0.0004s	2543.9		
233 bits	ecdh (nistb233)	0.0006s	1770.7		
283 bits	ecdh (nistb283)	0.0010s	999.0		
409 bits	ecdh (nistb409)	0.0018s	545.8		
571 bits	ecdh (nistb571)	0.0042s	239.3		
256 bits	ecdh (brainpoolP256r1)	0.0007s	1489.5		
256 bits	ecdh (brainpoolP256t1)	0.0007s	1444.1		
384 bits	ecdh (brainpoolP384r1)	0.0018s	550.1		
384 bits	ecdh (brainpoolP384t1)	0.0016s	630.3		
512 bits	ecdh (brainpoolP512r1)	0.0026s	377.9		
512 bits	ecdh (brainpoolP512t1)	0.0026s	389.7		
253 bits	ecdh (X25519)	0.0001s	13794.3		
448 bits	ecdh (X448)	0.0008s	1200.1		
		sign	verify	sign/s	verify/s
253 bits	EdDSA (Ed25519)	0.0001s	0.0002s	14717.6	5266.4
456 bits	EdDSA (Ed448)	0.0005s	0.0009s	1935.7	1064.9

Network Security

Computer Network Defence - 2023/24

Version 1.1 29/09/2023

We pivot to a more network centric focus in our defensive activities. This tutorial concentrates on packet processing. Understanding how information flows on a network, what is information is being transmitted, and to what endpoints is important for being able to secure it appropriately. With this kind of information we can then use it to inform how we manage the flow of information across our networks, with firewalls being one of the most common approaches to this. These in turn come with a challenge of how describe and effectively communicate the rules and policies - particularly to less network inclined users.

The tutorial is largely constructed as a series of steps to be followed, with a reduced content relating to open ended 'problem solving'. Your attention is directed to the various questions for you to consider as you work through the material. These serve to guide your further application of knowledge and provide you with a structure in which to practice your skills.

The practical tasks this week consist of four distinct groupings:

PACKETS [Task 1 - 3] Packets form the core of working with any kind of network security. These tasks provide an introduction to working with packets at the command line, rather than with GUI applications such as Wireshark. This is important as one starts processing data at scale. **These are important skills we will be relying on in the second half of the course.**

FIREWALLS [Task 4 & 5] The ability to concisely and accurately convey firewall rules and what action is taken on network packets is critical to successful network defense.

ENRICHMENT [Task 6] Raw packet data while plentiful, is often overwhelming. In addition it may only give insight to part of what is going on. These tasks look at some techniques for noise reduction, filtering, and how to go about enriching the data to allow you to gain more insight.

READING There are five brief news articles to review.,

These broad areas can all be worked on independently, with no inter-dependency between the sections. The order in which you approach these three areas is up to you. You are strongly encouraged to switch between tasks if you get stuck or find your concentration drifting.

You should further ensure that all tutorials are caught up by the start of the 7th week of lectures (after the upcoming reading week)

An issue that has been a source of frustration both for myself and students is in a number of cases where people failed to take heed of instructions to take snapshots or make backups. At this point in your academic career, making backups of your work (especially as cyber students where the domain exists to protect information) should be near automatic. Please pay attention to warnings in tutorial tasks indicating risky activity. These are in addition to good backup practice which you should undertake. You have large volumes of cloud storage available via OneDrive that you can make use of. Pay heed to the final concepts in this week's lecture.

The responsibility to ensure you have copies of important material rests with you.

This tut *may* be updated during the course of the week with additional support material, make sure you have the latest version as shown on Moodle.

Contents

1	Packet Analysis	1
2	Packet Analysis - tshark	3
3	Packet Analysis - tcpdump	8
4	Firewall Rules	12
5	Firewall Rules... again!	14
6	Filtering Noise	16
7	Deliverable	19
	Resources	19
A	News	20
B	Network Cheat Sheet	21
C	Syntax Reference	24
D	Simple Firewall Syntax	28

CHANGELOG

1. v1.0 - Initial Release
 2. v1.1 - Typo in Task 4 (12), and further warnings about tshark
-

1 Packet Analysis

You have been provided with a small dataset of packet data logged via a Network Telescope — a type of security sensor that has no active services and does not respond to traffic. This data is in pcap format. Note that destination addresses have been anonymised. As part of the processing, the packets have also been trimmed at 64 bytes to save space and remove any payloads present.

Open the data file and answer the following questions. For each, keep a detailed record of how you obtained the answer. These notes will be useful for you going forward. The files you need for this and the following two tasks can be found in the folder Tasks1-3 and are:

- `pcap-sample-anon.cap` Abridged data file (80KB). Use this for initial testing.
MD5: `ebc45d1e166575b21258e811b6a8c62b`
- `telescope2013_anon.cap` Complete datafile (100MB)
MD5: `e8003d52b0971e7ec863d396c70dfbc4`

Wireshark can be used for all the questions below (data file is small enough). The answer is not always immediately apparent, and you may need to do a little post processing of the data outside of Wireshark. Task 2 & 3 explore can process this from the command line and are a great way of validating results. Wireshark does not scale to doing this kind of analysis beyond a few 100 000 packets, or a few hundred MB of data. As rule of thumb you should have around 10x the memory available as the size of the packet capture you wish to load. This partially due to its protocol dissection functionality. This is seen as one of the biggest drawbacks of using Wireshark for exploratory analysis.



Seeing double!

The protocol dissection can also be over eager at times for both Wireshark and tshark (since they rely on the same underlying libraries). The most common case of this is when processing **ICMP Type 3** packets which contain the *headers* of the packet that generated the error. This may result in there being two source or destination addresses when you export data. This can apply in other cases as well. You must be aware of this issue and if using these tools, ensure that the output is suitably post processed, or an alternate tool like `tcpdump` used.



There is **no risk** in opening the data files on your local system, and you may feel more comfortable processing it on MacOS/Windows while you finish working through the Linux tutorials. You must be comfortable processing data on the command line.



Wireshark Installation

You will need to install Wireshark for this task. Should you choose to do it on your VM you can use the command:

```
$sudo apt install wireshark tshark
```

You may wish to install these tools on the host system if you need more memory available. You should not need >4GB of RAM to process the larger capture file. If you are doing this on the host is **strongly** recommended to shut your VM down to free up memory, before processing the data.

For Windows/MacOS systems you can install the tools from the software homepage if you do not have it installed already.

You are strongly encouraged to read and work through Task 2 and Task 3 **before** commencing with solving the questions in this activity. You will find these offer some alternate solutions, and most importantly a means of validating the results obtained.

Given your experience working with network packet data so far in your degree, it is expected that this task is attempted using Wireshark. While useful, it does **not** scale to larger datasets such as we will be working with. What is important is that you can obtain an answer. The combination of tools you used is very much an individual choice. There are many ways to solve these (one being to **avoid** Wireshark other than for detailed deep inspection)



Guiding Questions

Primarily use **Wireshark** to answer the following questions relating to the provided data file:

- (a) How many packets are there?
- (b) How many unique sources were observed?
- (c) How many unique destinations are observed?
- (d) What were the **top 10 targeted** ports for TCP and UDP? This is based on packet count.
- (e) For each of the ports from the list above, briefly state why you think they are being scanned.
- (f) What are the start and end times of the file?
- (g) What was the average Packets Per Second rate?
- (h) Plot a histogram of packet lengths.
- (i) Explain why you think there is one notable spike.
- (j) What traffic is responsible for the large packets? Why is this possible?
- (k) The top communications pairing is some 88 000 packets.
 - (i) What are the endpoints?
 - (ii) What protocol is being used?
 - (iii) Why do you think there is this high rate of communication?
- (l) A large spike of traffic occurred in the latter part of the data
 - (i) What was it?
 - (ii) When did it occur?

You may find some of this easier using alternate tools, or additional tools for doing some sorting and filtering. Wireshark is a very useful tool, but sometime not the best for the job.



Wireshark Resources

Some additional resources for working with tshark are:

- [Wireshark Manual - DisplayFilters](#)
- [Wireshark Tutorial: Display Filter Expressions](#)
- [Wireshark - IO Graphs](#)

2 Packet Analysis - tshark

Command line with tshark

tshark is a command line version of Wireshark, that uses the same underlying libraries, but lacks the (somewhat significant) overhead of the GUI. As a result it has some improved scalability and can be used in scripted processing of data. After reviewing the section, repeat your analysis using tshark as you did with Wireshark, using the provided questions as a guide.

The general syntax for the tshark command is:

```
tshark options or more usefully tshark -nr inputfile -Y filteroptions
```

These can be broken down as:

options A very large range of options are available as detailed in the manual page. The most important needed for our purposes are:

- n Disable name resolution. While it may make your output a little more sparse it can speed up processing significantly (by 10x or more!). It's always easier to do name resolution on final filtered data than at the start of the process.
- T<> Defines the format of output. *fields* is the most useful
- E<> Set options for output when -Tfields is defined. The *separator* option is the most useful when set as *separator=/t//s/<char>*. *<char>* is most usefully set to a comma , to produce csv output.
- e <field> Which field to emit when -Tfields selected. These are fields as used in the Wireshark display filters. A cheat sheet is given in Appendix B on page 27.
This is one of the single most powerful features with tshark for extracting single fields.
- q Quiet mode, which reduces the volume and verbosity of output.
- r Capture file to read from. If you want to read from *stdin* use -r - .
- w Capture file to write to. use - to write to *stdout*. This file is in *libpcap* format **not** text.
- Y <display filter> Packet display filter in Wireshark display filter used to filter traffic. The syntax is as per Appendix B on pages 27 and 28.

output tshark prints to *stdout* so can easily be chained with other commands or redirected to file.

The use of these options together with tshark is shown in the worked example below. You are strongly encouraged to refer to the system man page for the command for further details.

Worked Example

This is based on the file `pcap-sample-anon.cap` obtainable from Moodle as part of Task1-3.zip. This is a very small file. The following commands illustrate some very basic operation.

We can start by reading the file without any filtering. Note the use of the **-n** option.

```
$ tshark -nr pcap-sample-anon.cap
```

```
1  0.000000 193.46.254.169 → 192.168.168.226 TCP 60 42990 → 8000 [SYN] Seq=0 Win=1024 Len=0
2  0.152289 128.14.152.46 → 192.168.168.239 TCP 60 24176 → 443 [SYN] Seq=0 Win=1024 Len=0
3  0.388229 36.79.105.219 → 192.168.168.216 TCP 66 62754 → 1433 [SYN] Seq=0 Win=8192 Len=0
...
998 60.533851 179.43.157.139 → 192.168.168.126 TCP 60 43112 → 90 [SYN] Seq=0 Win=65535 Len=0
999 60.543657 193.46.254.169 → 192.168.168.237 TCP 60 42931 → 8085 [SYN] Seq=0 Win=1024 Len=0
1000 60.563951 205.185.125.73 → 192.168.168.85 TCP 60 41946 → 16163 [SYN] Seq=0 Win=65535 Len=0 MSS=53
```

While the lines seem to be numbered, we can repeat this and verify how many packets there are.

```
$ tshark -nr pcap-sample-anon.cap | wc -l
```

You should get a response of 1000 lines (and thus packets). Repeating this with a filter for ICMP traffic we get only 1 packet. Try with `tcp` and `udp`.


```
$ tshark -nr pcap-sample-anon.cap -Y icmp | wc -l
```

Filtering traffic by port (in this case 1433) allows us to filter down the traffic. Note the use of quotations around the filter string. Without this the tool will complain if you have more than a single filter term above. Both the == and eq notation are used for testing for the port match.

```
$ tshark -nr pcap-sample-anon.cap -Y "udp.port==1433 or tcp.port eq 1433"
  3   0.388229 36.79.105.219 → 192.168.168.216 TCP 66 62754 → 1433 [SYN] Seq=0 Win=8192 Len=0 MSS=1412
 22   1.862842 112.4.208.182 → 192.168.168.1 TCP 66 61326 → 1433 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS
 24   1.904703 36.79.105.219 → 192.168.168.97 TCP 66 50034 → 1433 [SYN] Seq=0 Win=8192 Len=0 MSS=1412 W
...
989  60.061259 112.4.208.182 → 192.168.168.225 TCP 66 [TCP Retransmission] 50630 → 1433 [SYN] Seq=0 Win
991  60.161875 36.79.105.219 → 192.168.168.111 TCP 66 65114 → 1433 [SYN] Seq=0 Win=8192 Len=0 MSS=1412
997  60.469297 36.79.105.219 → 192.168.168.222 TCP 66 [TCP Retransmission] 62850 → 1433 [SYN] Seq=0 Win
```

Finally we can write some traffic to a file for further processing.

```
$ tshark -nr pcap-sample-anon.cap -Y "udp.port==1433 or tcp.port eq 1433" -w pcap-sample-anon-port1433.
```

You can verify this is correct by checking is 11236 bytes in size, and should contain 112 lines if you count using `wc -l` on the textual output. **Note:** You will get a different hash when doing the same filtering with `tcpdump`. This is due to the way these applications write the pcap header. The files contents are identical and can be verified as such.

You should astutely notice from looking at the output of filtering for port 1433 above that there is one source network that appears to have more sources than the others. This can be isolated for further investigation.

```
$ tshark -nr pcap-sample-anon-port1433.cap -Y "ip.src == 112.4.0.0/16"
```

Output shows only a single host 112.4.208.182, and thus is not as interesting as it first seemed. Note the use of `ip.src` to explicitly match the source address. Exploring more lets look for any for any traffic targeting host 192.168.168.42 in the destination network.

```
$ tshark -nr pcap-sample-anon-port1433.cap -Y "ip.dst == 192.168.168.42"
 23  12.374121 112.4.208.182 → 192.168.168.42 TCP 66 55675 → 1433 [SYN] Seq=0 Win=8192 Len=0 MSS=1460
```



When working with complex filters its often easier to use intermediate files rather than keeping building more complex filters. This approach also saves having to reprocess datasets. This can be very important as larger datasets are used.

`tshark` requires one to jump though a number of hoops when working with time. By default it prints time relative to the start of the file. A simple approach to getting more human readable time is by explicitly printing fields we need. The following command prints the *timestamp* of each packet along with the *source* and *destination* IP addresses.

```
$ tshark -nr pcap-sample-anon-port1433.cap -T fields -e frame.time -e ip.src -e ip.dst
Apr  1, 2021 00:00:00.396189000 SAST 36.79.105.219 192.168.168.216
Apr  1, 2021 00:00:01.870802000 SAST 112.4.208.182 192.168.168.1
Apr  1, 2021 00:00:01.912663000 SAST 36.79.105.219 192.168.168.97
...
Apr  1, 2021 00:01:00.069219000 SAST 112.4.208.182 192.168.168.225
Apr  1, 2021 00:01:00.169835000 SAST 36.79.105.219 192.168.168.111
Apr  1, 2021 00:01:00.477257000 SAST 36.79.105.219 192.168.168.222
```

While quite human readable this is not ideal for programmatic processing on the command line. To get this in a format that is usable, an adjustment needs to be made to the command.

```
$ tshark -nr pcap-sample-anon-port1433.cap -t ad -T fields -e _ws.col.Time -e ip.src -e ip.dst
2021-04-01 00:00:00,396189 36.79.105.219 192.168.168.216
2021-04-01 00:00:01,870802 112.4.208.182 192.168.168.1
2021-04-01 00:00:01,912663 36.79.105.219 192.168.168.97
...
```

```
2021-04-01 00:01:00,069219 112.4.208.182 192.168.168.225
2021-04-01 00:01:00,169835 36.79.105.219 192.168.168.111
2021-04-01 00:01:00,477257 36.79.105.219 192.168.168.222
```



tshark time formats

You can list the various time formats that `tshark` support having passed to the `-t` parameter by running `tshark -t xxx`. the resulting error message provides a quick reference. Time such as **ad** is printed using local timezone. **ud** can be used for providing a human readable format relative to UTC. the examples below will be using this, take note of the date change.



Can you determine the timezone of the system this analysis was done on? From this could you deduce where it *could* be located geographically?

Often there is a need to transform pcap data into a csv file for easier processing by other tools. This can be done fairly easily. The following produces a csv file from `pcap-sample-anon-port1433.cap`.

```
$ tshark -nr pcap-sample-anon-port1433.cap -t ud -T fields -e _ws.col.Time -e ip.proto
-e ip.src -e ip.dst -e udp.srcport -e tcp.srcport -E separator=, -E header=y
```

```
_ws.col.Time,ip.proto,ip.src,ip.dst,udp.srcport,tcp.srcport
2021-03-31 22:00:00,396189,6,36.79.105.219,192.168.168.216,,62754
2021-03-31 22:00:01,870802,6,112.4.208.182,192.168.168.1,,61326
2021-03-31 22:00:01,912663,6,36.79.105.219,192.168.168.97,,50034
2021-03-31 22:00:01,923670,6,112.4.208.182,192.168.168.154,,61351
2021-03-31 22:00:02,122905,6,112.4.208.182,192.168.168.63,,61496
2021-03-31 22:00:02,213957,6,36.79.105.219,192.168.168.65,,50128
...
```

We note however that there is an issue with this as the date contains a comma as well. This may work well in some instances. We have also produced a header row which is always useful to describe what the columns are — especially in a case like this where `udp.srcport` is largely empty. This issue can be fixed by adding `-E quote=d` which will double quote (") values.

```
$ tshark -nr pcap-sample-anon-port1433.cap -t ud -T fields -e _ws.col.Time -e ip.proto
-e ip.src -e ip.dst -e udp.srcport -e tcp.srcport -E separator=, -E header=y -E quote=d
```

```
_ws.col.Time,ip.proto,ip.src,ip.dst,udp.srcport,tcp.srcport
"2021-03-31 22:00:00,396189","6","36.79.105.219","192.168.168.216",,"62754"
"2021-03-31 22:00:01,870802","6","112.4.208.182","192.168.168.1",,"61326"
"2021-03-31 22:00:01,912663","6","36.79.105.219","192.168.168.97",,"50034"
"2021-03-31 22:00:01,923670","6","112.4.208.182","192.168.168.154",,"61351"
...
"2021-03-31 22:00:59,960874","6","36.79.105.219","192.168.168.110",,"64441"
"2021-03-31 22:01:00,069219","6","112.4.208.182","192.168.168.225",,"50630"
"2021-03-31 22:01:00,169835","6","36.79.105.219","192.168.168.111",,"65114"
"2021-03-31 22:01:00,477257","6","36.79.105.219","192.168.168.222",,"62850"
```

This while slightly larger is quite unambiguous about any values that may contain the *separator* used. An alternate could have also been to change the *separator* to another value.

The above demonstrates some of the simpler ways of using `tshark`. You are encouraged to practice, in addition to trying to use `tshark` to repeat the tasks on page 2.

**BEWARE OF `tshark` FILTERS**

As previously noted, `tshark` tries very hard to decode packets. This includes decoding payload inside other packets such as ICMP datagrams, and where it *thinks* tunnelling protocols such as IPsec and GRE have been used. That can lead to some unexpected behaviour, particularly when using display filters. For example, while one would expect the following to only display UDP packets:

```
-Y "udp"
```

It will also display UDP packets as part of ICMP datagrams. When wanting to filter for TCP or UDP traffic only, it is advised to explicitly exclude ICMP using a filter such as:

```
-Y "udp and not icmp" or -Y "tcp and not icmp"
```

An alternative to this is to use `tcpdump` to filter the traffic to a temporary *pcapfile* and then use `tshark` for processing.

**`tshark` Resources**

Some additional resources for working with `tshark` are:

- [tshark tutorial and filter examples](#)
- [tshark: Basic Tutorial with Practical Examples](#)
- [tshark and Termshark tutorial: Capture and view wireshark captures in a console](#)
🕒 13:05
- [TShark - Basic Commands & Overview](#) 🕒 18:23

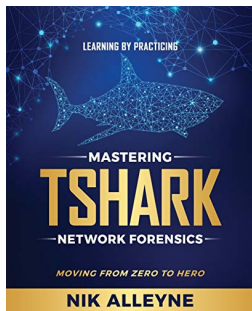
**Guiding Questions**

Use **`tshark`** to answer the following questions relating to the provided data file:

- How many packets are there?
- How many unique sources were observed?
- How many unique destinations are observed?
- What were the top 10 targeted ports for TCP and UDP?
- What are the start and end times of the file?
- What was the average Packets Per Second rate?
- Extract a list of packet lengths with the number of occurrences of each as a csv file.
- Can you isolate the traffic in the spike identified in the previous exercise?
- What traffic (as identified by srcip and ports) is responsible for the large packets? Why is this possible?
- Can you isolate and describe the traffic responsible for the 88 000 packets previously identified?
- Evaluate your answers against those you found with Wireshark. Did they differ? If so can you determine why and by how much?

**Dump info**

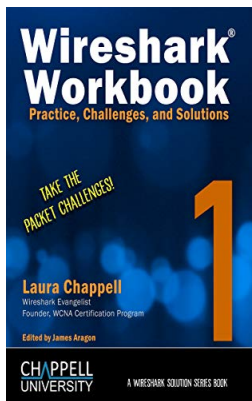
The `wireshark`/`tshark` packages install a number of other useful tools such as `capinfo`, `mergcap` and `editcap` which can be extremely useful when working with larger files and doing manipulation of data.



Learning by Practicing - Mastering TShark Network Forensics

Nik Alleyne June 1, 2020

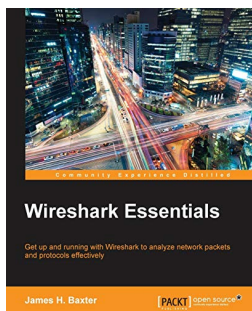
Mastering TShark Network Forensics, can be considered the definitive repository of practical TShark knowledge. It is your one-stop shop for all you need to master TShark, with adequate references to allow you to go deeper on peripheral topics if you so choose. While this book is written specifically for Network Forensics Analysts, it is equally beneficial to anyone who supports the network infrastructure. This means, Network Administrators, Security Specialists, Network Engineers, etc., will all benefit from this book.



Wireshark Workbook 1: Practice, Challenges, and Solutions

(A Wireshark Solution by *Laura Chappell* November 23, 2019

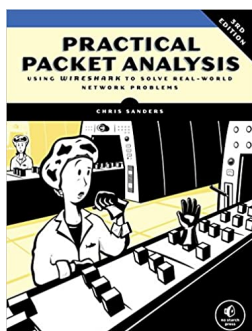
This Wireshark Workbook contains 16 sets of lab questions with fully-documented solutions to each question. It is designed to test your knowledge of Wireshark and TCP/IP analysis by focusing on your ability to locate answers to network traffic questions. Now you'll be able to see the step-by-step processes used to get the answer to those challenging questions.



Learn Wireshark: Confidently navigate the Wireshark interface and solve real-world networking problems

by *Lisa Bock* August 23, 2019

Learn Wireshark provides a solid overview of basic protocol analysis and helps you to navigate the Wireshark interface, so you can confidently examine common protocols such as TCP, IP, and ICMP. The book starts by outlining the benefits of traffic analysis, takes you through the evolution of Wireshark, and then covers the phases of packet analysis. We'll review some of the command line tools and outline how to download and install Wireshark on either a PC or MAC. You'll gain a better understanding of what happens when you tap into the data stream, and learn how to personalize the Wireshark interface.



Practical Packet Analysis, 3E: Using Wireshark to Solve Real-World Network Problems 3rd Ed

Chris Sanders March 30, 2017

Practical Packet Analysis will teach you to make sense of your packet captures so that you can better troubleshoot network problems. You'll find added coverage of IPv6 and SMTP, a new chapter on the powerful command line packet analyzers tcpdump and TShark, and an appendix on how to read and reference packet values using a packet map.

3 Packet Analysis - tcpdump

Command line with tcpdump

`tcpdump` is the original packet processing engine. While it has its limits it is relatively fast, scalable and can do some levels of decoding. Repeat your processing undertaken in the previous task using a combination of `tcpdump` along with other command-line tools like `sed`, `awk`, `uniq` and `sort`. Build on your skills and knowledge of text processing from earlier Tutorials to assist you. The general syntax for the `tcpdump` command is:

```
tcpdump options [-r|-w] file filterstring
```

These can be broken down as:

options A wide range of functionality is controlled by these. Those most important for our purposes are:

- n Disable name resolution. This applies at both Layer 3 and Layer 4. While it may make your output a little more sparse it can speed up processing significantly (by 10x or more!).
- c How many packets to process e.g. `-c 100`. You can think of this as the equivalent of the `head` command. Useful if you are only wanting to process a few to check parsing or output formatting is correct.
- t Defines the way timestamps are output. use multiple times to change output. Details are in the mail page. beware that this can slow down processing slightly so should be avoided if not needed.
- q Quiet mode, which reduces the volume and verbosity of output.
- v Verbose output. Adds information such as the time to live, identification, total length and options in an IP packet are printed. Also enables additional packet integrity checks such as verifying the IP and ICMP header checksum. can be used multiple times to increase output.
- r Capture file to read from. If you want to read from *stdin* use `-r -`.
- w Capture file to write to. This can be useful to do some initial filtering or reduction of a file and then save that traffic separately where it can be processed by a tool such as Wireshark or `tshark`.

file This is a file to be read from (`-r`) or written to (`-w`). This file is in *libpcap*¹ format. Its important to note this is not a text file, similarly `tcpdump` cannot process text input, such as that generated capturing *stdout* from another tool.

filterstring This is the string which controls the filtering of traffic. the BPF language is used which is very flexible, and also widely used across tools and vendors. Details of the language syntax can be found in the manual page for `pcap-filter` on your system. If this is *omitted* all packets in the file will be processed. Details can be found in Appendix B and a specific cheat sheet on page 26.

output capture you may want to capture the text based output from `tcpdump` for processing by other tools. This can be done using file redirection of *stdout* using `—` or `>` for file redirection.

The use of these components is shown in the worked example below. You are strongly encouraged to refer to the system man page for the command.



You should build up a saved list/text document of common command lines or partial argument/options lists for quick reference in the future.

¹ *libpcap* has been around since the early days of networking and is a standard format for the exchange of raw packet information across vendors, tools and transmission media. More can be found out about it at tcpdump.org and on the [WireShark Wiki](#)

Worked Example

This is based on the file `pcap-sample-anon.cap` obtainable from Moodle as part of Task1.zip. This is a very small file. The following commands illustrate some very basic operation.

We can start by reading the file without any filtering. Note the use of the `-n` option. This takes around 2 seconds. Omitting the disabling of resolution results in the process taking 2 minutes – so 60 times longer!

```
$ tcpdump -nr pcap-sample-anon.cap
```

We can repeat this and see how many packets there are.

```
$ tcpdump -nr pcap-sample-anon.cap | wc -l
```

You should get a response of 1000 lines (and thus packets). Repeating this with a filter for UDP traffic we get only 37 packets.

```
$ tcpdump -nr pcap-sample-anon.cap udp | wc -l
```

Filtering traffic by port (in this case 23) allows us to filter down the traffic. In this case note the use of `-q` and how it changes the output.

```
$ tcpdump -qnr pcap-sample-anon.cap port 23
reading from file pcap-sample-anon.cap, link-type EN10MB (Ethernet)
00:00:00.982444 IP 39.85.54.4.3538 > 192.168.168.63.23: tcp 0
00:00:06.552790 IP 98.7.202.212.8749 > 192.168.168.61.23: tcp 0
00:00:06.846959 IP 90.217.153.70.27722 > 192.168.168.235.23: tcp 0
...
00:00:59.054269 IP 223.80.100.131.46210 > 192.168.168.189.23: tcp 0
00:00:59.264875 IP 187.167.197.125.35130 > 192.168.168.63.23: tcp 0
00:01:00.326286 IP 1.228.210.164.16030 > 192.168.168.177.23: tcp 0
```

Looking at how we can apply what you have learned, we will now look at isolating some traffic involving port 22. The SSH service typically runs on port 22/tcp. Similar to the command above, we can now select traffic involving port 22 and write it to a file. Being able to extract smaller blocks of traffic for further investigation in another tool, is a very useful thing to be able to do. This is especially so when working with very large captures and tools such as Wireshark.

```
$ time tcpdump -qnr pcap-sample-anon.cap -w pcap-sample-anon-port22.cap port 22
```



Time

The `time` command can be placed in front of a new command or series of commands to see how long it takes. This is useful when you are trying to estimate how long a big job will take when processing a smaller sample, or if you are wanting performance data. It has **no** effect on the commands following.

Note the use of `time` preceding the command we ran. This results in output telling you how long a command took to execute. While useful for monitoring long running tasks it is also a great way to work out how long larger data processing is likely to take, after you have tried a run on a smaller dataset.

Use `tcpdump` to replay the file you created - `pcap-sample-anon-port22.cap`. You should astutely notice from looking at the output of filtering for port 22 above that there is one source network that appears to have more sources than the others. This can be isolated for further investigation.

```
$ tcpdump -qnr pcap-sample-anon.cap port 22 and src net 61.177.172.0/24
```

Output shows only a single host, and thus is not as interesting as it first seemed.

Exploring more lets look for any for any traffic targeting host .123 in the destination network.

```
$ tcpdump -qnr pcap-sample-anon.cap dst host 192.168.168.123
00:00:10.849480 IP 49.245.1.175.55555 > 192.168.168.123.23: tcp 0
00:00:12.671732 IP 185.195.26.17.3 > 192.168.168.123.12072: tcp 0
00:00:58.149866 IP 146.56.203.75.18207 > 192.168.168.123.23: tcp 0
```


Repeat the command above but using the **-t** flag to provide some timing information, and we see a single **-t** strips the time field. This is something worth remembering for the future.

```
IP 49.245.1.175.55555 > 192.168.168.123.23: tcp 0
IP 185.195.26.17.3 > 192.168.168.123.12072: tcp 0
IP 146.56.203.75.18207 > 192.168.168.123.23: tcp 0
```

Using **-tttt** provides more human readable timestamps, which are decoded in the form *YYYY-MM-DD hh:mm:ss.ms*. This is a lot easier to work with for text based processing than **-tt** which provides the time as an Epoch² stamp. The latter is however often much easier to work with programmatically – such as inputting the data into *pandas* or using the Python *datetime* functions, and is significantly more compact when storing in a database of any sorts!

```
2021-04-01 00:00:10.849480 IP 49.245.1.175.55555 > 192.168.168.123.23: tcp 0
2021-04-01 00:00:12.671732 IP 185.195.26.17.3 > 192.168.168.123.12072: tcp 0
2021-04-01 00:00:58.149866 IP 146.56.203.75.18207 > 192.168.168.123.23: tcp 0
```

The above demonstrates some of the simpler ways of using *tcpdump*. You are encouraged to practice, in addition to trying to use *tcpdump* to repeat the tasks on page 2.



Trust, but Verify

It is always a *good idea* to validate output and analysis using different tools. This is especially so if you are looking at using it as evidence in legal processing or as report of network abuse.



Guiding Questions

Use **tcpdump** to answer the following questions relating to the provided data file:

- (a) How many packets are there?
- (b) How many unique sources were observed?
- (c) How many unique destinations are observed?
- (d) What were the top 10 targeted ports for TCP and UDP?
- (e) What are the start and end times of the file?
- (f) What was the average Packets Per Second rate?
- (g) Extract a list of packet lengths with the number of occurrences of each as a csv file.
- (h) Can you isolate the traffic in the spike identified in the previous exercises ?
- (i) What traffic (as identified by srcip and ports) is responsible for the large packets? Why is this possible?
- (j) Can you isolate and describe the traffic responsible for the 88 000 packets previously identified?
- (k) Evaluate your answers against those you found with Wireshark, and *tshark*. Did they differ? If so can you determine why and by how much?
- (l) What were the relative speeds of processing by the tools?



Check your answers.

A self evaluation for these questions will be available on Moodle from 09h00 on Wednesday.

²The world started on 00:00 UTC 1 January 1970 as far as unix time is concerned, and the [unix epoch timestamp](#) consists of the number of seconds since then. A handy [converter](#) is available but for bulk data you need to do thing pragmatically.



1. What would the advantage be of using command line processing over Wireshark be?
2. Can you combine your commands into a single script that runs them from start to finish for a given file and print out (or saves) the results?



tcpdump Resources

Some other resources you might find useful are:

- Daniel Miessler's [A tcpdump Tutorial with Examples — 50 Ways to Isolate Traffic](#).
- [Tcpdump advanced filters](#) by Sebastien Wains
- Julia Evans provides a 'graphic novel' style exploration of tcpdump, entitled [Let's learn tcpdump!](#)
- A more verbose discussion about [BPF filters](#)



Faster Processing

tracereport is a *very* fast and useful tool that can be installed to make this kind of analysis much faster using the command: `sudo apt install libtrace-tools`.

The `-m` flag produces a good summary output in the file `misc.rpt`.

This is an example of a more specialist tool. There are several other tools that come with this similarly based of libtrace originally developed by the [WAND Research Group](#) in New Zealand. The example tools are quite useful, but for real speed and power one needs to use the library directly (in C/Rust/Python) - something outside the scope of this course.

You can easily enumerate what these are by typing `trace` at the command prompt, followed by hitting `tab` to see what the completion options are in the `bash` shell.

A resource worth noting is these tools use the concept of *uri* to define the type of data they are reading or writing. Refer to the [URI formats](#) wiki. You can read more about the design in their [original paper](#).

4 Firewall Rules

An important task when dealing with firewalls is to be able to describe firewall rules in a concise written form. There are many differing formats and syntax's this can be done with. For the purposes of this course, we will use a simplified format of the syntax used by FreeBSD for it's `ipfw(8)` command. The simplified (reduced) syntax we will be using is contained in Appendix D on Page 28.



You may Assume the following configuration for your network:

- For addressing the external subnet of your firewall is 185.23.45.64/29
- Your internal LAN is 178.34.10.0/24 and is a full publicly accessible netblock.
- The internal gateway for the LAN is .1

Using the information above, complete rules for the items 1-13 listed below. Pay attention to these instructions:

- Start by writing these as **stateless** rules therefor accounting for traffic in both directions. Assume you have a default **deny** rule.
- Review the work you have done above, and re-write these now using **stateful firewalling**. Use the option `keep-state` on rules you wish to track. The action `check-state` can be used to check state.
- You should have **two** rulesets at the end of this process.
- Compare them for readability as well as the total number of rules generated.

1. Block all traffic going though the firewall
2. Allow a single Internal admin host to access the firewall ssh port and the statistics web interface running on port 9098/tcp
3. Deny internal hosts access to all hosts in the **range** 196.23.167.1-127
HINT: how could you optimise this ?
4. Allow LAN hosts to make SSH connections to the Internet
5. Implement a filter on the outside interface of your firewall for all [RFC1918](#) Traffic.
6. Extend your filter above to include all IPv4 sources defined in [RFC3330](#).
7. You have a 3rd 'leg' on your firewall with the network 172.19.41.0/24. Set up rules to allow your internal (LAN) network to connect to all services on this netblock. Only mysql connections should be allowed back to the host .241 on the LAN.
8. Restrict access from internal hosts to the internet to only ports 22, SMTPS and IMAPS.
9. Restrict internet bound traffic so that hosts have to make use of the installed web proxy server. Assume your proxy server runs on a separate IP of .87 on your LAN.
10. You have two internal DNS servers, These run on .2 and .3 on your lan. Ensure that these are the only hosts that can be used to query dns on the internet.
11. Allow INBOUND traffic to port 31337/tcp on the external interface. This should be redirected to port 3389/tcp or port 22/tcp on the LAN IP addresses of .65 and .34 respectively.
12. Add logging to the rules you made in tasks 7 above (the mysql back connect). This should not replace the rules above that you have previously implemented.
*HINT: The **log** action differs from the others in that a packet does not exit if a rule only logs.*
13. Adopt a posture for ICMP packets coming from the internet in which you accept and log pings, but deny everything else. [ICMP Types](#) are what you should look for.

- **Make sure you have these carefully written down each of the above as individual responses.**
- Combine your individual responses into into a reasonable single rule-set that could viably be loaded on a host. **Save your ruleset!**



It is important to remember that direction of traffic as when using `in` or `out` is relative to the firewall. Rule numbering indicates the order that they are processed in. Avoid having rules with the same number. Having unique numbers can often make logging much more useful.



You may find this easier to start with by think of the rules in complete isolation. When you start combining things it becomes a little more complicated. it is **always** good practice to have your default rule explicitly stated (often done as rule 65535 — which is the highest numbered rule supported, and thus last processed should there be not other matches.



Rule numbering

Unless one is needing to have very large or complex rulesets, its always a good idea to 'leave space' for insertion later. Typical convention numbers rules by 10's or 100's. This allows for the insertion of other rules at a later point. This is very much less of an issue on more modern GUI based firewall system. It still makes sense however when planning rulesets.



Swap your rulesets with someone else in the class, and consider the following:

- Are they able to describe what you are intending?
- Can you interpret their rulesets?
- How do you think you could both improve your ruleset writing.
- Challenge each other to write some other rules.

5 Firewall Rules...again!

Previously you have translated from textual descriptions of firewall rules to a fairly simple formal syntax. This task, does the inverse, 'decompiling' the syntax to a more 'human readable' format like more suitable for non network staff. The firewall rule-set below is a fairly simplified firewall script used on a host. The driver for this is `/bin/sh` which is a simpler variant of the `bash` shell you are used to.



Reading rules

The firewall ruleset below is implemented using a shell script. you should be familiar with the syntax used. The `$cmd` at the start of each line runs the appropriate command to load each rule as defined at the start of the script. Regard `$pif` as the **external** firewall interface. Inspect it and describe what is being done with each group of rules (as separated by spaces, and labelled as `#Qx – Q1 to Q16`).



ipfw?

The `ipfw` tool is used on FreeBSD systems as one of the ways to manage firewalls. It is well regarded as having a readable syntax. and we are using a reduced/simplified version of its syntax in this course. Rules are all numberes and are processed in sequential order from lowest to highest. A packet exits the rule chain on the **first** matching rule that it encounters.

```
#!/bin/sh
# Flush out the list before we begin.
ipfw -q -f flush
# We now have an empty ruleset
# Set rules command prefix
cmd="ipfw -q add"
pif="vtnet0"      # interface name of NIC attached to Internet
#Q1
$cmd 00010 allow all from any to any via lo0
#Q2
$cmd 00101 check-state
#Q3
$cmd 00120 allow log udp from any to any 67 out via $pif keep-state
#Q4
$cmd 00130 allow tcp from me to any dst-port 53 out via $pif setup keep-state
$cmd 00131 allow udp from me to any dst-port 53 out via $pif keep-state
#Q5
$cmd 00200 allow tcp from any to any 80 out via $pif setup keep-state
$cmd 00220 allow tcp from any to any 443 out via $pif setup keep-state
#Q6
$cmd 00230 allow tcp from any to any 25 out via $pif setup keep-state
$cmd 00231 allow tcp from any to any 110 out via $pif setup keep-state
$cmd 00232 allow tcp from any to any 465 out via $pif setup keep-state
$cmd 00233 allow tcp from any to any 587 out via $pif setup keep-state
#Q7
$cmd 00250 allow icmp from any to any out via $pif keep-state
#Q8
$cmd 00260 allow udp from any to any 123 out via $pif setup keep-state
#Q9
$cmd 00299 deny log all from any to any out via $pif
#Q10
$cmd 00300 deny all from 192.168.0.0/16 to any in via $pif
$cmd 00301 deny all from 172.16.0.0/12 to any in via $pif
$cmd 00302 deny all from 10.0.0.0/8 to any in via $pif
$cmd 00303 deny all from 127.0.0.0/8 to any in via $pif
$cmd 00304 deny all from 0.0.0.0/8 to any in via $pif
```

```
$cmd 00305 deny all from 169.254.0.0/16 to any in via $pif
$cmd 00306 deny all from 192.0.2.0/24 to any in via $pif
$cmd 00307 deny all from 204.152.64.0/23 to any in via $pif
$cmd 00308 deny all from 224.0.0.0/3 to any in via $pif
#Q11
$cmd 00310 deny icmp from any to any in via $pif
#Q12
$cmd 00315 deny tcp from any to any 113 in via $pif
#Q13
$cmd 00320 deny tcp from any to any 137 in via $pif
$cmd 00321 deny tcp from any to any 138 in via $pif
$cmd 00322 deny tcp from any to any 139 in via $pif
$cmd 00323 deny tcp from any to any 81 in via $pif
#Q14
$cmd 00332 deny tcp from any to any established in via $pif
#Q15
$cmd 00400 allow tcp from any to me 80 in via $pif
#Q16
$cmd 00410 allow tcp from any to me 22 in via $pif setup limit src-addr 2
$cmd 00430 allow udp from any 53 to me in via $pif
$cmd 00431 allow tcp from any 53 to me in via $pif

# Reject and log all other incoming connections
$cmd 00499 deny log all from any to any in via $pif
```



Having worked through the firewall ruleset above and added annotated it appropriately, answer the following questions:

1. What services are running on the host with this firewall ruleset.
2. Who 'owns' the netblock 204.152.64.0/23
3. Why do you think this is firewalled?
4. What is is normally used for?



Me me me me

In the ruleset above you will notice the use of the `me` keyword where one would expect an address. This is a fairly common concept to most firewalls and is a shorthand to refer to the IP address of **any interface** on the firewall. This can significantly shorten rulesets in practice, but can be confusing if you do not keep a clear mental picture of the setup and network topology. If you find it easier, you can use explicit rules.

6 Filtering Noise

This task builds on the type of packet capture dataset you dealt with in Task 1. The data for this task comes from September 2019 and has been somewhat pre-filtered and a list of unique source addresses provided for you, along with the number of packets seen (see `telescope0919_anon.uniqsrcip.csv`). You have also been provided with a list of actors tracked by GREYNOISE.IO. GREYNOISE.IO tracks what are deemed to be **benign** scanning activities from research organisations. This data was current at the time the traffic was captured: (`greynoise.actors.json` and `greynoise.csv`).

You will find the files you need in the `tut6` folder:

- `telescope0919_anon.uniqsrcip.csv` – a csv file with the fields: `srcip`, packet count
- `greynoise.actors.json` and `greynoise.csv` - GREYNOISE.IO actor data.
 - The csv file has been converted from the json for your convenience.
 - They contain the same information.
 - The `greynoise.csv` file has the structure of "organisation", followed by a comma separated list of IP addresses attributable/in use by that organisation.
- `shodan.txt` – extract of shodan.io data from the `greynoise.csv`.

NOTE: The json file is provided for completeness or for those that want to use Python, where it may be easier to iterate over. A conversion from json to usable (plain) text can also be done using the `jq` tool. Both of these approaches are however beyond the scope of this Tutorial.

Shodan extraction

See `shodan.txt` for an extract of SHODAN's attributable IP addresses. You can use this list to determine how many of the SHODAN attributable IP addresses appeared in the sample network traffic. This data has already been extracted using the following command to convert it to a more useful form:

```
fgrep -i shodan greynoise.csv | sed -e 's/"/"/\g' | grep -vi shodan > shodan.txt
```

This is just one approach to processing the data. You can use this or a similar approach for extracting out the other GREYNOISE sources you need to process, replacing the relevant parts of the command. This is much easier than manually splitting and editing the file, which is likely to be error prone.



Getting Started

There are many ways to do this including using Microsoft Excel or other spreadsheet, however these are all fairly long involved processes. This is significantly faster on the *nix command line, or via custom Python Script. Consider reviewing the BASH tutorial on how to loop through file contents.

Don't Panic

This task may look rather daunting, but much of the hard work has already been done for you:

- The initial 2.3 GB pcap has been processed down to 18MB of **text** using the following command³:

```
time cat telescope0919_anon.cap.gz | tcpdump -nr - \
| awk '{print $3}' | awk -F. '{print $1"."$2"."$3"."$4}' \
| sort | uniq -c | sort -rn \
| awk '{print $2","$2}' > telescope0919_anon.uniqsrcip.csv
```

³What the `\?` – the `\` at the end of each line indicates to `bash` that the command string is not yet complete and so should not be executed. This is a much easier way to type and format commands than very long command lines - especially in scripts. An added bonus is that it makes them readable on a page where you cannot scroll.

The output of this is saved to file and you can see appropriate timing⁴ information:

```
real 2m43,736s
user 3m42,389s
sys 0m8,131s
```

The core of this task is about using loops to process (iterate over) your data – i.e. the same process with differing input variables. While it is strongly suggested to use the **command line** tools covered in the course thus far, or Python (see the `json`, and `csv` modules - but these are not needed). You can solve this in Excel as well (VLOOKUP and/or Pivot Tables may well be helpful), but may find the process a lot more cumbersome (and of limited scale with larger datasets).

The Shodan task (c) should be used as a test case. Get this working **first**. Once you can get this working, the remainder of (e) becomes a case of repeating what you have done.



Exploring Data

Using the information provided in the files from greynoise.io, you need to process the provided network traffic set (`telescope0919_anon.uniqrscip.csv`) and determine how many hosts, and packets etc. are attributable to each of the 51 organisations listed by greynoise.io. This is based on the IP addresses listed for them being observed as source addresses.

You are required to answer the following, showing/explaining your workings as to how you arrived at your answer:

- (a) How many unique sources were observed?
- (b) How many packets were observed?
- (c) How many packets were attributable to Shodan.io?
- (d) What percentage of total packets was this?
- (e) For **each** remaining actor in the `greynoise.csv`
 - (i) Determine how many hosts were observed
 - (ii) How many packets were observed?
- (f) Express the value above as a percentage of the total packets recorded
- (g) Draw a plot of the actors sorted by % of total packets, ordered in decreasing order. The image must contain your student email address.
- (h) In total how many packets and hosts (and as a % of the total) in the scan data can be deemed to be benign (not potentially malicious) based on the GREYNOISE.IO data?
- (i) Briefly reflect on what you have found, and discuss it with other Students Does this benign scanning constitute a significant portion of traffic?
- (j) Why do you think it is important that traffic can be labelled like this when we are considering it from a network security perspective?
- (k) Why do you think these organisations conduct active scanning of the Internet?

Ensure you save your answers.



Check your answers.

A self evaluation will be available on Moodle from 09h00 on Wednesday.

⁴Run time 163s on an i3, with magnet drives. Timing with smaller samples can give you a good idea of how long processing will take on larger datasets.



What to do?

You are to use the information provided in the greynoise files, to do some filtering (and ultimately enrichment) of the data contained in the "telescope0919_anon.uniqrsrcip.csv". You should split the greynoise data into the different listed organisations listed. Each organisation has a number of IP Addresses listed that are associated with them. Use this information to answer the questions.

The activity of filtering the traffic data to check for the presence of IP addresses for each organisation is quite repetitive. As such, this is the perfect case for a for a loop. The processing does not take very long (at the command line). This should give you separate output files you can use to answer **e** onwards.

Activities **a-d** are to get the process prepared. Both **a** and **b** apply to *all traffic*. Activities **c** and **d** apply to traffic that is attributable to the IP addresses listed in the shodan.txt (ie Operated by shodan based on greynoise's research).

Activities in **e** replicates **c** and **d** but for the other organisations listed in the greynoise data.



Plotting is hard?

Plotting data is not particularly difficult once one has is correctly formatted. Getting data appropriately summarised and formatted for the tool of choice is often the much harder task. Unfortunately there are few cases where you can just give data to an existing application and expect it to work flawlessly. This is a particularly true of large datasets.

Once your data is summarised and suitably formatted (a csv is a common choice) you have a wide range of applications to choose from. These include:

- Spreadsheet applications such as Microsoft Excel
- Python libraries such as *matplotlib*
- Specialist analysis tools such as R and *ggplot* - although this is a **very** steep learning curve, but wow is the output beautiful!
- Online tools such as Google Charts or plot.ly - the later's online [chart studio](#) is rather useful.
- Other specialist graphing tools like [gnuplot](#)

7 Deliverable

While the **entire** tutorial contents are important the following specific items are in scope for forthcoming engagement activities.

Packet Analysis

You required to have answered the Questions (a)-(l) in Task 1. This may be much easier if you approach the analysis having been through Tasks 2 and 3 as well.

What is important is that you can obtain an answer. The combination of tools you used is very much a individual choice. There are many ways to solve these (one being to avoid `wireshark` other than for detailed deep inspection).



If you are using `tshark` for processing be very sure you verify output at the various intermediate steps and ensure you are cleaning up the data appropriately.

You should be comfortable doing basic filtering and processing with `tcpdump` and `tshark`.

The answers to the questions are in scope for **Engagement Activity 3**.

Firewall rules

You should have the list of 13 independent rule-sets from Task 4. You should be comfortable reading and writing simple rule-sets such as that provided.

The **stateless** responses are in scope for **Engagement Activity 2**

From Task 5 you should have answered the four questions on page 15.

These are in scope for **Engagement Activity 3**

Exploring Data

You should have completed tasks (a) - (h) on page 17.

These are in scope for **Engagement Activity 2**

A News

The following are new articles of interest in the last week. These are all relatively short, and you should get in the habit of staying aware of what is going on in the industry at large. The topics discussed here fall in scope for the engagement tasks.



News Week of 24 September 2023

DDoS [Pro-Russia hacker group NoName launched a DDoS attack on Canadian airports causing severe disruptions](#)

The Canada Border Services Agency (CBSA) finally confirmed on Tuesday that “connectivity issues that affected kiosks and electronic gates at airports” are the result of a distributed denial of service (DDoS) attack.”

BREACH [TransUnion Denies Breach After Hacker Publishes Allegedly Stolen Data](#)

TransUnion’s announcement comes two days after a threat actor published on a cybercrime forum a database allegedly containing the information of roughly 60K individuals.

RESEARCH [GOLD MELODY: Profile of an Initial Access Broker](#)

The GOLD MELODY threat group acts as an initial access broker (IAB) that sells access to compromised organizations for other cybercriminals to exploit. This financially motivated group has been active since at least 2017. The victimology suggests opportunistic attacks for financial gain rather a targeted campaign conducted by a state-sponsored threat group for espionage, destruction, or disruption.

ATTACK [Cybercriminals Exploit the Moroccan Tragedy in New Scam Campaign](#)

This blog post exposes a scam that has taken advantage of the earthquake in Morocco by deceiving users to buy relief equipment purportedly meant to aid quake victims.

MALWARE [Malware Attacking MS-SQL Servers](#)

Numerous variants of Gh0st RAT are often used in attacks targeting MS-SQL servers. AhnLab Security Emergency response Center (ASEC) monitors attacks targeting poorly managed MS-SQL servers and publishes quarterly statistics through the ASEC Reports.

These articles are in scope for Engagement Activity 2.

B Network Cheat Sheet

For reference the packet headers⁵ for IP, TCP, UDP and ICMP are shown. These are useful to refer to when you are building up your syntax strings.

IPv4 Header

Version	IHL	ToS	Total Length	
Identification			Flgs	Fragment Offset
Time To Live	Protocol		Header Checksum	
Source Address				
Destination Address				
Options				Padding

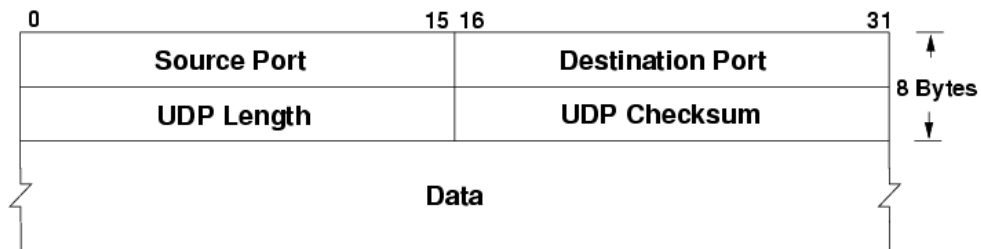
TCP Header

32 Bit																																
0																	15	16														31
source port																destination port																
sequence number																																
acknowledgment number																																
data offset		reserved		C	E	U	A	P	R	S	F	window																				
				W	R	E	G	K	H	T	N	I																				
checksum																urgent pointer																
options (0 oder mehr 32-Bit-Wörter)																																
data (Nutzdaten)																																

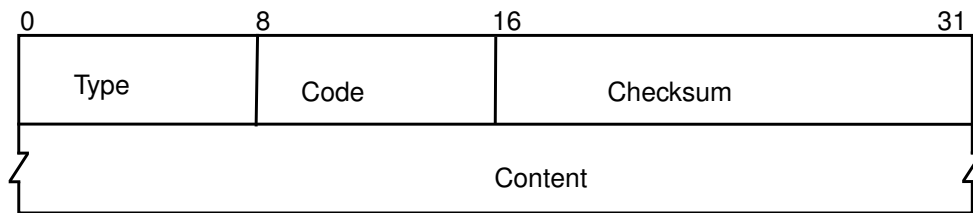
 TCP-Header

⁵These images are all Mro, [CC BY-SA 3.0](#), via Wikimedia Commons

UDP Header



ICMP Header



IP Subnetting

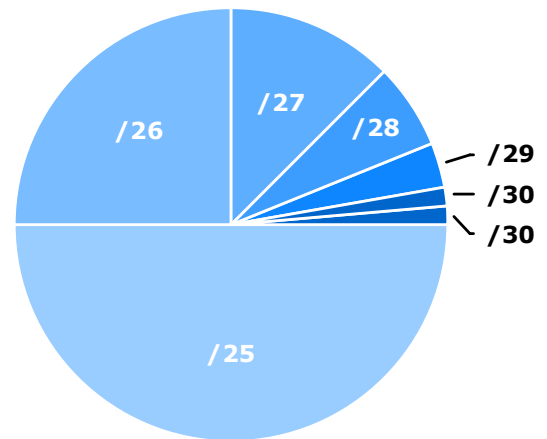
The following page contains a quick reference sheet developed by Jeremy Stretch to remind you how IPv4 subnet addressing works. This is very useful when working with filters. Full sized [PDF](#).

IPv4 SUBNETTING

packetlife.net

Subnets				Decimal to Binary			
CIDR	Subnet Mask	Addresses	Wildcard	Subnet Mask	Wildcard		
/32	255.255.255.255	1	0.0.0.0	255	1111	1111	0 0000 0000
/31	255.255.255.254	2	0.0.0.1	254	1111	1110	1 0000 0001
/30	255.255.255.252	4	0.0.0.3	252	1111	1100	3 0000 0011
/29	255.255.255.248	8	0.0.0.7	248	1111	1000	7 0000 0111
/28	255.255.255.240	16	0.0.0.15	240	1111	0000	15 0000 1111
/27	255.255.255.224	32	0.0.0.31	224	1110	0000	31 0001 1111
/26	255.255.255.192	64	0.0.0.63	192	1100	0000	63 0011 1111
/25	255.255.255.128	128	0.0.0.127	128	1000	0000	127 0111 1111
/24	255.255.255.0	256	0.0.0.255	0	0000	0000	255 1111 1111
/23	255.255.254.0	512	0.0.1.255				
/22	255.255.252.0	1,024	0.0.3.255				
/21	255.255.248.0	2,048	0.0.7.255				
/20	255.255.240.0	4,096	0.0.15.255				
/19	255.255.224.0	8,192	0.0.31.255				
/18	255.255.192.0	16,384	0.0.63.255				
/17	255.255.128.0	32,768	0.0.127.255				
/16	255.255.0.0	65,536	0.0.255.255				
/15	255.254.0.0	131,072	0.1.255.255				
/14	255.252.0.0	262,144	0.3.255.255				
/13	255.248.0.0	524,288	0.7.255.255				
/12	255.240.0.0	1,048,576	0.15.255.255				
/11	255.224.0.0	2,097,152	0.31.255.255				
/10	255.192.0.0	4,194,304	0.63.255.255				
/9	255.128.0.0	8,388,608	0.127.255.255				
/8	255.0.0.0	16,777,216	0.255.255.255				
/7	254.0.0.0	33,554,432	1.255.255.255				
/6	252.0.0.0	67,108,864	3.255.255.255				
/5	248.0.0.0	134,217,728	7.255.255.255				
/4	240.0.0.0	268,435,456	15.255.255.255				
/3	224.0.0.0	536,870,912	31.255.255.255				
/2	192.0.0.0	1,073,741,824	63.255.255.255				
/1	128.0.0.0	2,147,483,648	127.255.255.255				
/0	0.0.0.0	4,294,967,296	255.255.255.255				

Subnet Proportion



Classful Ranges

- A** 0.0.0.0 – 127.255.255.255
- B** 128.0.0.0 – 191.255.255.255
- C** 192.0.0.0 – 223.255.255.255
- D** 224.0.0.0 – 239.255.255.255
- E** 240.0.0.0 – 255.255.255.255

Reserved Ranges

- RFC 1918** 10.0.0.0 – 10.255.255.255
- Localhost** 127.0.0.0 – 127.255.255.255
- RFC 1918** 172.16.0.0 – 172.31.255.255
- RFC 1918** 192.168.0.0 – 192.168.255.255

Terminology

CIDR

Classless interdomain routing was developed to provide more granularity than legacy classful addressing; CIDR notation is expressed as /XX

VLSM

Variable-length subnet masks are an arbitrary length between 0 and 32 bits; CIDR relies on VLSMs to define routes

C Syntax Reference

The following pages contain useful cheat sheets that have developed by Jeremy Stretch for:

- BPF for `tcpdump` and other tools that use the BPF syntax. Full sized [PDF](#).
- Wireshark filters (as used by `tshark` and `wireshark`). Full sized [PDF](#).

You may want to print off a copy for easy reference while you work.

TCPDUMP

packetlife.net

Command Line Options					
-A	Print frame payload in ASCII	-q	Quick output		
-c <count>	Exit after capturing count packets	-r <file>	Read packets from file		
-D	List available interfaces	-s <len>	Capture up to len bytes per packet		
-e	Print link-level headers	-S	Print absolute TCP sequence numbers		
-F <file>	Use file as the filter expression	-t	Don't print timestamps		
-G <n>	Rotate the dump file every n seconds	-v[v[v]]	Print more verbose output		
-i <iface>	Specifies the capture interface	-w <file>	Write captured packets to file		
-K	Don't verify TCP checksums	-x	Print frame payload in hex		
-L	List data link types for the interface	-X	Print frame payload in hex and ASCII		
-n	Don't convert addresses to names	-y <type>	Specify the data link type		
-p	Don't capture in promiscuous mode	-Z <user>	Drop privileges from root to user		
Capture Filter Primitives					
[src dst] host <host>		Matches a host as the IP source, destination, or either			
ether [src dst] host <ehost>		Matches a host as the Ethernet source, destination, or either			
gateway host <host>		Matches packets which used host as a gateway			
[src dst] net <network>/<len>		Matches packets to or from an endpoint residing in network			
[tcp udp] [src dst] port <port>		Matches TCP or UDP packets sent to/from port			
[tcp udp] [src dst] portrange <p1>-<p2>		Matches TCP or UDP packets to/from a port in the given range			
less <length>		Matches packets less than or equal to length			
greater <length>		Matches packets greater than or equal to length			
(ether ip ip6) proto <protocol>		Matches an Ethernet, IPv4, or IPv6 protocol			
(ether ip) broadcast		Matches Ethernet or IPv4 broadcasts			
(ether ip ip6) multicast		Matches Ethernet, IPv4, or IPv6 multicasts			
type (mgt ctl data) [subtype <subtype>]		Matches 802.11 frames based on type and optional subtype			
vlan [<vlan>]		Matches 802.1Q frames, optionally with a VLAN ID of vlan			
mpls [<label>]		Matches MPLS packets, optionally with a label of label			
<expr> <relop> <expr>		Matches packets by an arbitrary expression			
Protocols			Modifiers	Examples	
arp	ip6	slip	! or not	udp dst port not 53	UDP not bound for port 53
ether	link	tcp	&& or and	host 10.0.0.1 && host 10.0.0.2	Traffic between these hosts
fddi	ppp	tr	or or	tcp dst port 80 or 8080	Packets to either TCP port
icmp	radio	udp	ICMP Types		
ip	rarp	wlan	icmp-echoreply	icmp-routeradvert	icmp-tstampreply
TCP Flags			icmp-unreach	icmp-routersolicit	icmp-ireq
tcp-urg	tcp-rst		icmp-sourcequench	icmp-timxceed	icmp-ireqreply
tcp-ack	tcp-syn		icmp-redirect	icmp-paramprob	icmp-maskreq
tcp-psh	tcp-fin		icmp-echo	icmp-tstamp	icmp-maskreply

by Jeremy Stretch

v2.0

WIRESHARK DISPLAY FILTERS · PART 1 packetlife.net

Ethernet		
eth.addr	eth.len	eth.src
eth.dst	eth.lg	eth.trailer
eth.ig	eth.multicast	eth.type
IEEE 802.1Q		
vlan.cfi	vlan.id	vlan.priority
vlan.etype	vlan.len	vlan.trailer
IPv4		
ip.addr	ip.fragment.overlap.conflict	
ip.checksum	ip.fragment.toolongfragment	
ip.checksum_bad	ip.fragments	
ip.checksum_good	ip.hdr_len	
ip.dsfield	ip.host	
ip.dsfield.ce	ip.id	
ip.dsfield.dscp	ip.len	
ip.dsfield.ect	ip.proto	
ip.dst	ip.reassembled_in	
ip.dst_host	ip.src	
ip.flags	ip.src_host	
ip.flags.df	ip.tos	
ip.flags.mf	ip.tos.cost	
ip.flags.rb	ip.tos.delay	
ip.frag_offset	ip.tos.precedence	
ip.fragment	ip.tos.reliability	
ip.fragment.error	ip.tos.throughput	
ip.fragment.multipletails	ip.ttl	
ip.fragment.overlap	ip.version	
IPv6		
ipv6.addr	ipv6.hop_opt	
ipv6.class	ipv6.host	
ipv6.dst	ipv6.mipv6_home_address	
ipv6.dst_host	ipv6.mipv6_length	
ipv6.dst_opt	ipv6.mipv6_type	
ipv6.flow	ipv6.nxt	
ipv6.fragment	ipv6.opt.pad1	
ipv6.fragment.error	ipv6.opt.padn	
ipv6.fragment.more	ipv6.plen	
ipv6.fragment.multipletails	ipv6.reassembled_in	
ipv6.fragment.offset	ipv6.routing_hdr	
ipv6.fragment.overlap	ipv6.routing_hdr.addr	
ipv6.fragment.overlap.conflict	ipv6.routing_hdr.left	
ipv6.fragment.toolongfragment	ipv6.routing_hdr.type	
ipv6.fragments	ipv6.src	
ipv6.fragment.id	ipv6.src_host	
ipv6.hlim	ipv6.version	
ARP		
arp.dst.hw_mac	arp.proto.size	
arp.dst.proto_ipv4	arp.proto.type	
arp.hw.size	arp.src.hw_mac	
arp.hw.type	arp.src.proto_ipv4	
arp.opcode		
TCP		
tcp.ack	tcp.options.qs	
tcp.checksum	tcp.options.sack	
tcp.checksum_bad	tcp.options.sack_le	
tcp.checksum_good	tcp.options.sack_perm	
tcp.continuation_to	tcp.options.sack_re	
tcp.dstport	tcp.options.time_stamp	
tcp.flags	tcp.options.wscale	
tcp.flags.ack	tcp.options.wscale_val	
tcp.flags.cwr	tcp.pdu.last_frame	
tcp.flags.ecn	tcp.pdu.size	
tcp.flags.fin	tcp.pdu.time	
tcp.flags.push	tcp.port	
tcp.flags.reset	tcp.reassembled_in	
tcp.flags.syn	tcp.segment	
tcp.flags.urg	tcp.segment.error	
tcp.hdr_len	tcp.segment.multipletails	
tcp.len	tcp.segment.overlap	
tcp.nxtseq	tcp.segment.overlap.conflict	
tcp.options	tcp.segment.toolongfragment	
tcp.options.cc	tcp.segments	
tcp.options.ccecho	tcp.seq	
tcp.options.ccnew	tcp.srcport	
tcp.options.echo	tcp.time_delta	
tcp.options.echo_reply	tcp.time_relative	
tcp.options.md5	tcp.urgent_pointer	
tcp.options.mss	tcp.window_size	
tcp.options.mss_val		
UDP		
udp.checksum	udp.dstport	udp.srcport
udp.checksum_bad	udp.length	
udp.checksum_good	udp.port	
Operators		Logic
eq or ==		and or && Logical AND
ne or !=		or or Logical OR
gt or >		xor or ^^ Logical XOR
lt or <		not or ! Logical NOT
ge or >=		[n] [...] Substring operator
le or <=		

by Jeremy Stretch

v2.0

WIRESHARK DISPLAY FILTERS - PART 2 packetlife.net

Frame Relay			ICMPv6	
fr.becn	fr.de		icmpv6.all_comp	icmpv6.option.name_type.fqdn
fr.chdlctype	fr.dlci		icmpv6.checksum	icmpv6.option.name_x501
fr.control	fr.dlcore_control		icmpv6.checksum_bad	icmpv6.option.rsa.key_hash
fr.control.f	fr.ea		icmpv6.code	icmpv6.option.type
fr.control.ftype	fr.fecn		icmpv6.comp	icmpv6.ra.cur_hop_limit
fr.control.n_r	fr.lower_dlci		icmpv6.haad.ha_addrs	icmpv6.ra.reachable_time
fr.control.n_s	fr.nlpid		icmpv6.identifier	icmpv6.ra.retrans_timer
fr.control.p	fr.second_dlci		icmpv6.option	icmpv6.ra.router_lifetime
fr.control.s_ftype	fr.snap.oui		icmpv6.option.cga	icmpv6.recursive_dns_serv
fr.control.u_modifier_cmd	fr.snap.pid		icmpv6.option.length	icmpv6.type
fr.control.u_modifier_resp	fr.snaptype		icmpv6.option.name_type	
fr.cr	fr.third_dlci			
fr.dc	fr.upper_dlci			
PPP			RIP	
ppp.address	ppp.direction		rip.auth.passwd	rip.ip
ppp.control	ppp.protocol		rip.auth.type	rip.metric
			rip.command	rip.netmask
			rip.family	rip.next_hop
			rip.route_tag	rip.routing_domain
			rip.version	
MPLS			BGP	
mpls.bottom	mpls.oam.defect_location		bgp.aggregator_as	bgp.mp_reach_nlri_ipv4_prefix
mpls.cw.control	mpls.oam.defect_type		bgp.aggregator_origin	bgp.mp_unreach_nlri_ipv4_prefix
mpls.cw.res	mpls.oam.frequency		bgp.as_path	bgp.multi_exit_disc
mpls.exp	mpls.oam.function_type		bgp.cluster_identifier	bgp.next_hop
mpls.label	mpls.oam.ttsi		bgp.cluster_list	bgp.nlri_prefix
mpls.oam.bip16	mpls.ttl		bgp.community_as	bgp.origin
			bgp.community_value	bgp.originator_id
			bgp.local_pref	bgp.type
			bgp.mp_nlri_tnl_id	bgp.withdrawn_prefix
ICMP			HTTP	
icmp.checksum	icmp.ident	icmp.seq	http.accept	http.proxy_authorization
icmp.checksum_bad	icmp.mtu	icmp.type	http.accept_encoding	http.proxy_connect_host
icmp.code	icmp.redir_gw		http.accept_language	http.proxy_connect_port
DTP			http.authbasic	http.referer
dtp.neighbor	dtp.tlv_type	vtp.neighbor	http.authorization	http.request
dtp.tlv_len	dtp.version		http.cache_control	http.request.method
VTP			http.connection	http.request.uri
vtp.code	vtp.vlan_info.802_10_index		http.content_encoding	http.request.version
vtp.conf_rev_num	vtp.vlan_info.isl_vlan_id		http.content_length	http.response
vtp.followers	vtp.vlan_info.len		http.content_type	http.response.code
vtp.md	vtp.vlan_info.mtu_size		http.cookie	http.server
vtp.md5_digest	vtp.vlan_info.status.vlan_susp		http.date	http.set_cookie
vtp.md_len	vtp.vlan_info.tlv_len		http.host	http.transfer_encoding
vtp.seq_num	vtp.vlan_info.tlv_type		http.last_modified	http.user_agent
vtp.start_value	vtp.vlan_info.vlan_name		http.location	http.www_authenticate
vtp.upd_id	vtp.vlan_info.vlan_name_len		http.notification	http.x_forwarded_for
vtp.upd_ts	vtp.vlan_info.vlan_type		http.proxy_authenticate	
vtp.version				

by Jeremy Stretch

v2.0

D Simple Firewall Syntax

When creating an firewall rule, keywords must be written in the following order. Some keywords are mandatory while other keywords are optional. The words shown in uppercase represent a variable and the words shown in lowercase must precede the variable that follows it. The # symbol is used to mark the start of a comment and may appear at the end of a rule or on its own line. Blank lines are ignored. The overall format is:

```
RULE_NUMBER ACTION log LOG_AMOUNT PROTO from SRC SRC_PORT to DST DST_PORT OPTIONS
```

Unless otherwise specified, a *minimal* set of instructions can be used which follow the format of:

```
RULE_NUMBER ACTION PROTO from SRC SRC_PORT to DST DST_PORT
```

For example:

```
100 allow UDP from any to 8.8.8.8 53
```

```
200 allow TCP from 146.231.123.120/21 5127,3119 to 9.9.9.0/24 8127,672
```

This section provides an overview of these keywords and their options. It is not an exhaustive list of every possible option. Refer to ipfw(8) for a complete description of the rule syntax that can be used when creating IPFW rules.

RULE_NUMBER Each rule is associated with a number from 1 to 65534. The number is used to indicate the order of rule processing. Multiple rules can have the same number, in which case they are applied according to the order in which they have been added. Otherwise rules are processed top to bottom in numerical order. The packet exits (stops being processed) on the **first** match.

ACTION A rule can be associated with one of the following actions. The specified action will be executed when the packet matches the selection criterion of the rule.

- *allow/accept/pass/permit*: these keywords are equivalent and allow packets that match the rule.
- *check-state*: checks the packet against the dynamic state table. If a match is found, execute the action associated with the rule which generated this dynamic rule, otherwise move to the next rule. A check-state rule does not have selection criterion. If no check-state rule is present in the ruleset, the dynamic rules table is checked at the first keep-state or limit rule.
- *count*: updates counters for all packets that match the rule. The search continues with the next rule.
- *deny/drop*: either word silently discards packets that match this rule.

LOG_AMOUNT When a packet matches a rule with the **log** keyword, a message will be logged. Logging only occurs if the number of packets logged for that particular rule does not exceed a specified LOG_AMOUNT. If no LOG_AMOUNT is specified, the limit is taken from the default value.

PROTO This optional value can be used to specify any protocol name or number found in /etc/protocols. Commonly used are TCP, UDP, ICMP, but could include others such as GRE (47), ESP (50), AH (51) see [List of IP protocol Numbers](#) for details.

SRC The from keyword must be followed by the source address or a keyword that represents the source address. An address can be represented by any, me (any address configured on an interface on this system), When specifying an IP address, it can be optionally followed by its CIDR mask or subnet mask. For example, 1.2.3.4/25 or 1.2.3.4:255.255.255.128.

SRC_PORT An optional source port can be specified using the port number or name from /etc/services.

DST The to keyword must be followed by the destination address or a keyword that represents the destination address. The same keywords and addresses described in the SRC section can be used to describe the destination.

DST_PORT An optional destination port can be specified using the port number or name from /etc/services.

OPTIONS Several keywords can follow the source and destination. As the name suggests, OPTIONS are optional. Commonly used options include in or out, which specify the direction of packet flow, icmp types followed by the type of ICMP message, and keep-state.

Network Security II

Computer Network Defence - 2023/24

Version 1.02 16/10/2023

Continuing with the theme of network focused processing, this week explores the processing of network data further. Building on the initial processing of network data, enrichment (involving merging-in in and correlating with other information) is introduced. Being able to take raw event information and add additional context though iterating with other sources is an important part of the Cyber Threat Intelligence (CTI) process. In addition some elements of Auditing are introduced.

The tutorial is largely constructed as a series of steps to be followed, with a reduced content relating to open ended 'problem solving'. Your attention is directed to the various questions for you to consider as you work through the material. These serve to guide your further application of knowledge and provide you with a structure in which to practice your skills.

The practical tasks this week consist of three distinct groupings:

PACKETS [*Task 1 - 3*] These tasks build on your prior experience with working with packets at the command line, rather than with GUI applications such as Wireshark. This is important as one starts processing data at scale. Task 2 while quite long in terms of page count, is a series of steps for you to follow with plentiful example output. The final task of this group builds on the preceding two as well as the skills you developed in the previous tutorial.

HARDENING [*Task 4*] Returning to system tuning, this task explores some of the most notable *tunables* within the Linux kernel as they relate to networking and networking security.

LOGS & AUDITING [*Tasks 5 & 6*] System auditing both for integrity as well as looking for signs of certain actions is an important part of overall network security. These two tasks are intended to provide an overview of some of the processes/skill needed in order to do this. These are not intended to be time consuming problem solving exercises.

These broad areas can all be worked on independently, with no inter-dependency between the sections. The order in which you approach these three areas is up to you. You are strongly encouraged to switch between tasks if you get stuck or find your concentration drifting.

This tut *may* be updated during the course of the week with additional support material, make sure you have the latest version as shown on Moodle.

Contents

1	Enriching Data	1
2	Information Finding	6
3	Packets Revisited	17
4	Network Tuning	21
5	Linux Logging	24
6	Auditing	25
7	Deliverable	30
	Resources	30
A	News	31
B	NS Notes	32

CHANGELOG

- ➦ v1.0 - Initial Release
 - ➦ 1.01 - Fixed filename in Task 3
 - ➦ 1.02 - Fixed chmod command in logwatch in Task 6, and links in Linux Audit Cheatsheets
-

1 Enriching Data

IP addresses which we are able to gather as the result of logging or appropriate sensors are only part of the overall picture. We can use these as a basis for doing data enrichment to gain more knowledge about the IP addresses, and potentially some insight into any Actors behind them. The field of Threat Intelligence is an entire domain in its own right however it has become increasingly easy to provide intelligent enrichment of security data in recent years.

Geolocation

Geolocation as it relates to IP addresses (the typical data we work with) involved attribution of an IP address to a geographic location. This is different from the geolocation commonly used on your mobile device which can make use of additional information such as WIFI networks, cellular data and ultimately a GPS unit. IP geolocation relies on the use of a database containing information. These are maintained by a number of different organisations. One of the challenges is the accuracy of the databases. As a general rule, Country level attribution is very accurate, and this decreases as one moves to City and ultimately specific geographic co-ordinates.

Two different approaches to IP geolocation are that of using local, and online services. You should repeat this exercise using **both** methods.



Geolocation Accuracy

Geolocation accuracy is something one should always take into account, especially as it relates to attribution. Two resources on evaluating such IP databases as we are using are

- Shavitt, Y. and Zilberman, N., 2011. A geolocation databases study. IEEE Journal on Selected Areas in Communications, 29(10), pp.2044-2056. [doi:10.1109/JSAC.2011.111214](https://doi.org/10.1109/JSAC.2011.111214) Alternate
- Worley, J., 2018. [IP Geolocation: The Good, The Bad, & The Frustrating.](#)

Local Database

This approach uses an installed database on your system to query against. This process is usually quite fast, especially if you use a program to do the queries directly with the database.

One of the simplest approaches is to make use of the `geoiplookup` command in your VM. This can be installed by running `sudo apt install geoip-bin`. This installs the base toolkit and a version of the local database.

An example of this output is:

```
$ geoiplookup 146.231.23.123
GeoIP Country Edition: ZA, South Africa
```

Here we are given the two letter ISO country code (ZA) and the full country name. An alternate database that can be used that has a particularly good Python API is that provided by Maxmind,- particularly its GEOLITE2 FREE database offering.

- <https://dev.maxmind.com/geoip/geoip2/geolite2/>
- <https://pypi.org/project/GeoIP/>
- <https://pypi.org/project/python-geoip/>

Online Service

There are many online services providing geolocation for a single IP via a web page. These are difficult to script and do not all necessarily have an API you can use pragmatically. One service that is well used in the Information Security community is that offered by [TEAM CYMRU](https://www.team-cymru.com/), via `whois`¹. Have a look at the following additional resources.

- <https://www.team-cymru.com/IP-ASN-mapping.html>
- <https://team-cymru.com/community-services/ip-asn-mapping/#whois>
- <https://asn.cymru.com/>



It is very important that you do your best to reduce load on these kinds of online services especially those providing it for free. A very easy way of doing this is to make sure you only query each IP address once. Make sure you reduce your dataset appropriately.



The `cymruwhois` Python module works very well with their service but can be a little tricky to set up.

Working with Cymru Whois

Using the Whois interface requires a little preparation and can be done completely from the command line. With no programming required. The steps below are based on the [documentation](#) provided by TEAM CYMRU. This provides detailed steps you **must** follow.



Repeatedly sending incorrectly formatted data can result in your IP address being banned.

Ensure that your format is correct before sending large queries.

To be able to use the service, we first need a list of IP Addresses. It is important to note that *IP addresses* are required **not** *hostnames*. An example starting list:

```
151.101.67.5
151.101.3.5
151.101.195.5
151.101.131.5
193.35.52.51
23.13.34.7
195.88.55.16
195.88.54.16
```

This list now needs to be formatted correctly to be able to be processed by the service. The requirements for this are:

1. Instructions should start with `begin` on a line of its own
2. This *may* be followed by the term `verbose` if more information is required.
3. One IP address per line, ensure no duplicates.
4. A final command of `end`
5. It is good practice to have a blank line at the end

¹`whois` is a legacy protocol that is however very fast, efficient, and most importantly easily scriptable. More information can be found at <https://en.wikipedia.org/wiki/WHOIS> and in [RFC3912](#).

The easiest approach is to edit the file directly, however it is possible to construct a file pragmatically using a construct similar to the following:

```
$ echo begin >> cymru.query
$ echo verbose >> cymru.query
$ cat ip.list >> cymru.query
$ echo end >> cymru.query
```

Take note of the use of the >> operator which **appends** to a file rather than overwriting. After following the instructions above, the list of IP addresses we started with should look like:

```
begin
verbose
151.101.67.5
151.101.3.5
151.101.195.5
151.101.131.5
193.35.52.51
23.13.34.7
195.88.55.16
195.88.54.16
end
```

With this in place, we can now run the query using the `nc` (netcat) command to allow us to pipe our prepared command file to the whois service:

```
$ cat cymru.query | nc whois.cymru.com 43 > cymru.verbose
```

Looking at the contents of this file we are able to see a number of columns are returned:

```
$ cat cymru.verbose
Bulk mode; whois.cymru.com [2022-10-21 06:14:40 +0000]
54113 | 151.101.67.5 | 151.101.64.0/22 | US | arin | 2016-02-01 | FASTLY, US
54113 | 151.101.3.5 | 151.101.0.0/22 | US | arin | 2016-02-01 | FASTLY, US
54113 | 151.101.195.5 | 151.101.192.0/22 | US | arin | 2016-02-01 | FASTLY, US
54113 | 151.101.131.5 | 151.101.128.0/22 | US | arin | 2016-02-01 | FASTLY, US
58302 | 193.35.52.51 | 193.35.52.0/22 | NO | ripencc | 2012-06-26 | SAMFUNDET-AS, NO
16625 | 23.13.34.7 | 23.13.32.0/20 | US | arin | 2010-12-17 | AKAMAI-AS, US
39029 | 195.88.55.16 | 195.88.54.0/23 | NO | ripencc | 2009-02-13 | REDPILL-LINPRO Redpill
39029 | 195.88.54.16 | 195.88.54.0/23 | NO | ripencc | 2009-02-13 | REDPILL-LINPRO Redpill
```

The columns here are:

1. ASN
2. IP address (this is what you queries)
3. Netblock which IP belongs to
4. Country that netblock is attributed to
5. RIR responsible for the netblock
6. Date that netblock was allocated
7. Netblock owner organisation, with country. Note this country may not always match the country code returned for the ip especially for multinational organisations.

Running the query **without** the *verbose* command produces the following:

```
Bulk mode; whois.cymru.com [2022-10-21 06:16:25 +0000]
54113 | 151.101.67.5 | FASTLY, US
54113 | 151.101.3.5 | FASTLY, US
54113 | 151.101.195.5 | FASTLY, US
54113 | 151.101.131.5 | FASTLY, US
58302 | 193.35.52.51 | SAMFUNDET-AS, NO
```

16625	23.13.34.7	AKAMAI-AS, US
39029	195.88.55.16	REDPILL-LINPRO Redpill Linpro, NO
39029	195.88.54.16	REDPILL-LINPRO Redpill Linpro, NO

When using this service, you should choose what is best for your enrichment needs. Before processing large volumes of data, make sure you can format your input correctly, and are getting the output you want. As a general rule its best to keep your queries at around 50-100K IP/request - these will typically take between 40 and 90 seconds to run. Larger lists can be easily split up with the `split` command. See its manual page for details. These can then be individually processed and the results recombined.



Processing tips

- Pre-process your data, only query what you need.
- Remove duplicates.
- **Do not abuse the service, or you will find your IP blocked.** This is could include issuing multiple requests that are invalid.
- As guide towards minimum, keep your queries under 100K lines long Processing speed is on average around 1300/second, but this may vary. Not using verbose is around 50% faster. You may be able to do larger queries, faster, but be sensible, particularly if you are having timeout problems.
- Be aware that queries take time, a full query of the 834K entries takes 10-15 minutes. You can monitor the progress of this by running the command `watch wc -l filename` - where *filename* is the output file you are creating. This will update the count every 2 seconds.
- If you are having problems, keep your number of requests (a request can have multiple addresses in it) under 1000/hour.
- If you are getting timeouts, break your queries up in to smaller chunks 10 000 - 20 0000 rows of input. you can recombine data later.
- Check your query files carefully before submitting. Errors can cause problems later on -especially in scripts.

Application



Geolocation Enrichment

Use your list of IP addresses used in last weeks tutorial Task 6 (telescope0919_anon.uniqsrcip.csv). Your task is to enrich this data with country and other related information.

You should repeat these task using both local (offline) and online methods so you can compare their strengths and weaknesses.

What is and is not possible with each method. Briefly document your enrichment process, and answer the following:

- (a) Enrich the data. How long does it take to perform the enrichment? (Use the `time` command before your script/command).
- (b) What are the top 10 countries (by total number of IP addresses), and what % of the total do each these represent?
- (c) What are the top 10 ASN (by total number of IP addresses **you** observe in the data), and what % of the total do each these represent?
- (d) What are the top 10 CIDR Netblocks in which the observed IP addresses are located? Rank these by the number of IP addresses observed.
- (e) Produce a histogram/plot for each of the following (in descending order). You should calculate and use a % as the 'y' value to plot rather than actual count. Ignore any values less than 1% (of the total).
 - Countries
 - ASN
 - Netblocks



The TEAM CYMRU Whois service provides much more enrichment data than just geolocation. The additional information provided can provide substantial other insight when analysing incidents (and ultimately) securing networks and systems.

2 WHOIS, NSlookup, Host and Dig

WHOIS is a protocol defined for use to extract information about network information from various registries. Typically, this relates to Autonomous System Numbers (ASN), Netblock allocations (IP ranges - both IPv4 and IPv6) and DNS domain names. Pre-GDPR and similar regulations, there was substantially more information available. However, it still provides a valuable, fast means of gaining information.

Tools commonly used alongside `whois` are `nslookup`, `host` and `dig` which allow more detailed querying of the domain name system (which these days includes much more than just domain information – typically due to the overloading of the TXT records).



You can find out more about the command line options and the tools on a Unix system by trying to use the `-h` or `--help` flags. To get more detail you can use the `man` command to pull up the manual page

Your task today is to familiarise yourself with these tools on the **VM** platform. It is important to know how these tools work, before using many of the more automated data gathering frameworks. These tools often break, or fail to deal with unexpected results, and its valuable to be able to debug them, and/or verify their output. It is also important to be able to do information gathering in environments where you do not have access to additional tools.

DNS Tools: `dig` and `nslookup`

Before starting, ensure that the `dnsutils` package is installed on your system. These tools may already be installed as part of the `bind9-dnsutils` package.

How to use `dig`

The `dig` syntax consists of the hostname/IP address, name, and record as follows: `dig [HOSTNAME] [NAME] [RECORD]`

- Hostname consists of the domain name or IP address of the server.
- Name is DNS (Domain Name Server) of the server to query.
- Type is the DNS record to retrieve (default A is retrieved if not specified). Other known records are A, MX, TXT, CNAME, and NS.

The usage is pretty simple. We will start by understanding this command's output by querying the Noroff domain, as shown below.

```
>>> DiG 9.18.1-1ubuntu1.2-Ubuntu <<<> noroff.no
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8683
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;noroff.no.                IN      A

;; ANSWER SECTION:
noroff.no.                 5       IN      A      194.63.248.52

;; Query time: 3 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Fri Oct 20 06:54:47 UTC 2022
;; MSG SIZE rcvd: 54
```

This response consists of four areas you need to take note of:

- The header consists of information related to the dig version, status, id, and other additional information.
- The question area consists of the query you request while executing the dig command.
- The answer section is the important part where you can see the information about the query you have asked.
- The footer area consists of the information, date, and message size.

The important takeaway here is the line after ; ; AUTHORITY SECTION: . This provides us with the information that `noroff.no` has an address (A record) of `194.63.248.52`.

This may seem like a lot of information that is not all immediately useful, but the origin of `dig` is as a diagnostic tool. Fortunately it provides a rich set of options and you can use these to return information that is more useful directly. The `+short` option returns the 'short' version of the response only. In the second case below, the A (address) record is specifically being requested.

```
$ dig +short noroff.no
194.63.248.52
$ dig +short noroff.no A
194.63.248.52
```

Using the record specification, we can ask for specific types of information. The NS record in DNS is used to record the nameservers for the domain. Again using Noroff as an example:

```
$ dig +short noroff.no NS
ns3.hyp.net.
ns1.hyp.net.
ns2.hyp.net.
```

Similarly we can look for information on mailservers which are indicated by the Mail exchanger (MX) record. `dig` is relatively forgiving in the ordering of its options - far more so than other tools.

```
$ dig noroff.no MX +short
10 noroff-no.mail.protection.outlook.com.
```

By default, `dig` commands will query the name servers listed in `/etc/resolv.conf` to perform a DNS lookup for you. If you want to change the default behaviour, specify the hostname or IP address of the name server after the `@` symbol. The below command will send the DNS query to the Google name server, and look for TXT records associated with the domain.

```
$ dig @8.8.8.8 noroff.no TXT +short
"facebook-domain-verification=475ngd8hnpbgocr4wwd4z8mm5vlatm"
"v=spf1 ip4:85.10.255.106 ip4:80.239.56.122 ... include:_spf.salesforce.com -all"
"pardot577561=774df8bcef66d7d4b1e467dcd9695d08f940f9e638558c792aac25b7fd27084e"
"atlassian-domain-verification=KnQ3n9I9ZZ/LvdG/6cYD3bD5u/fKs9rrdWZ0Q9GhVavQki0VGC0Gxdv9Id90Ggl1"
"MS=ms57824509"
"facebook-domain-verification=fypuoe7rwc3nofgdjgiukj9k24ozf2"
```

Using nslookup

You can use the `nslookup` command to query the name server for various domain records, as shown below.

```
$ nslookup noroff.no
Server:      127.0.0.53
Address:     127.0.0.53#53
```

```
Non-authoritative answer:
Name:   noroff.no
Address: 194.63.248.52
Name:   noroff.no
Address: 2a01:5b40:0:248::52
```

The lines describe what is being returned. An important point to pay attention to is the Server: and following address line. This is the information of the server that answered your query.

Looking up the nameservers for the Noroff domain, provides similar but more verbose output than dig.

```
$ nslookup -type=NS noroff.no
Server:      127.0.0.53
Address:     127.0.0.53#53
```

Non-authoritative answer:

```
noroff.no      nameserver = ns1.hyp.net.
noroff.no      nameserver = ns2.hyp.net.
noroff.no      nameserver = ns3.hyp.net.
```

Authoritative answers can be found from:

```
ns1.hyp.net    internet address = 151.249.124.1
ns1.hyp.net    has AAAA address 2a01:5b40:ac1::1
ns2.hyp.net    internet address = 192.174.68.10
ns2.hyp.net    has AAAA address 2001:67c:1bc::10
ns3.hyp.net    internet address = 151.249.126.3
ns3.hyp.net    has AAAA address 2a01:5b40:ac3::1
```

Similarly other record types can be requested. Looking up just an A record we see a difference to the first example, in that there is no IPv6 record returned. this is because we have asked specifically for an A (IPv4 address), rather than AAAA for IPv6.

```
$ nslookup -type=A noroff.no
Server:      127.0.0.53
Address:     127.0.0.53#53
```

Non-authoritative answer:

```
Name:   noroff.no
Address: 194.63.248.52
```

To repeat the example shown in dig requesting TXT records form a different server, can be done using the following:

```
$ nslookup -type=TXT noroff.no 8.8.8.8
;; Truncated, retrying in TCP mode.
Server:      8.8.8.8
Address:     8.8.8.8#53
```

Non-authoritative answer:

```
noroff.no      text = "facebook-domain-verification=475ngd8hnpbgecr4wwd4z8mm5vlatm"
noroff.no      text = "v=spf1 ip4:85.10.255.1..... include:_spf.salesforce.com -all"
noroff.no      text = "pardot577561=77....95d08f940f9e638558c792aac25b7fd27084e"
noroff.no      text = "atlassian-domain-verification=KnQ3n9....9GhVavQki0VGC0Gxdv9Id90Ggl1"
noroff.no      text = "MS=ms57824509"
noroff.no      text = "facebook-domain-verification=fypuoe7rwc3nofgdjgiukj9k24ozf2"
```

Authoritative answers can be found from:

Whois

The `whois` command has many useful options that can be used for exploring internet related names and addressing information. Before following along with the examples below, ensure that `whois` is installed. This can be done using `apt install whois` on your system. This tool allows you to query about internet related objects, the most common of which are addresses, domains and ASNs

You can use `whois` to find out more about an IP address. To find out more about the IP address of the Noroff webserver (194.63.248.52) we can use the following:

```
$ whois 194.63.248.52
% This is the RIPE Database query service.
% The objects are in RPSL format.
%
% The RIPE Database is subject to Terms and Conditions.
% See http://www.ripe.net/db/support/db-terms-conditions.pdf

% Note: this output has been filtered.
%       To receive output for a database update, use the "-B" flag.

% Information related to '194.63.248.0 - 194.63.255.255'

% Abuse contact for '194.63.248.0 - 194.63.255.255' is 'abuse@domeneshop.no'

inetnum:        194.63.248.0 - 194.63.255.255
netname:        NO-DOMENESHOP
org:            ORG-DA421-RIPE
country:        NO
admin-c:        SS19786-RIPE
tech-c:         HH2777-RIPE
status:         ASSIGNED PI
mnt-by:         RIPE-NCC-END-MNT
mnt-by:         DOMENESHOP-MNT
mnt-routes:     DOMENESHOP-MNT
mnt-domains:    DOMENESHOP-MNT
created:        1970-01-01T00:00:00Z
last-modified:  2016-04-14T09:55:14Z
source:         RIPE # Filtered

organisation:   ORG-DA421-RIPE
org-name:       Domeneshop AS
country:        NO
org-type:       LIR
address:        Christian Krohgs gate 16
address:        0186
address:        Oslo
address:        NORWAY
phone:          +4722943333
fax-no:         +4722943334
mnt-ref:        DOMENESHOP-MNT
mnt-ref:        RIPE-NCC-HM-MNT
mnt-by:         RIPE-NCC-HM-MNT
mnt-by:         DOMENESHOP-MNT
abuse-c:        HH2777-RIPE
created:        2012-06-07T11:55:20Z
last-modified:  2020-12-16T13:26:59Z
source:         RIPE # Filtered
admin-c:        SS19786-RIPE

role:           Domeneshop Hostmaster
address:        Domeneshop AS
```

```

address:      Christian Krohgs gate 16
address:      0186 Oslo
address:      Norway
phone:        +47 22 94 33 33
fax-no:       +47 22 94 33 34
abuse-mailbox: abuse@domeneshop.no
admin-c:      SS19786-RIPE
tech-c:       SS19786-RIPE
nic-hdl:      HH2777-RIPE
mnt-by:       DOMENESHOP-MNT
created:      1970-01-01T00:00:00Z
last-modified: 2021-03-06T09:54:48Z
source:       RIPE # Filtered

```

```

person:       Stale Schumacher
address:      Domeneshop AS
address:      Christian Krohgs gate 16
address:      0186 Oslo
address:      Norway
phone:        +47 22 94 33 33
fax-no:       +47 22 94 33 34
nic-hdl:      SS19786-RIPE
mnt-by:       DOMENESHOP-MNT
created:      2012-06-07T12:47:02Z
last-modified: 2021-03-06T09:53:34Z
source:       RIPE # Filtered

```

% Information related to '194.63.248.0/24AS12996'

```

route:        194.63.248.0/24
descr:        DOMENESHOP
origin:       AS12996
mnt-by:       DOMENESHOP-MNT
created:      2021-11-21T11:39:58Z
last-modified: 2021-11-24T09:45:28Z
source:       RIPE

```

% This query was served by the RIPE Database Query Service version 1.104 (WAGYU)

This produces a volume of information, but it is intended to be able to be filtered with standard text-processing tools. Compare this to the more specialist whois services like that offered by Team Cymru, which provides commonly used information in a more concise format.

```

$ whois -h whois.cymru.com 194.63.248.52
AS      | IP              | AS Name
12996   | 194.63.248.52   | DOMENESHOP Oslo, Norway, NO

```

More verbose output is available. It is worth noting that it is much more efficient to use the bulk query mode for this service described elsewhere in the tutorial.

```

$ whois -h whois.cymru.com -v 194.63.248.52
Warning: RIPE flags used with a traditional server.
AS      | IP              | BGP Prefix      | CC | Registry | Allocated | AS Name
12996   | 194.63.248.52   | 194.63.248.0/24 | NO | ripencc   | 1999-11-09 | DOMENESHOP Oslo, Norway, NO

```

We can use whois to find out more about the AS12996 listed in the response above. While generally of interest for networking engineers, the information provide can also be sued to help determine contacts in the case of abuse, or even what range of addresses to block.

```

$ whois AS12996
% This is the RIPE Database query service.

```

```
% The objects are in RPSL format.
%
% The RIPE Database is subject to Terms and Conditions.
% See http://www.ripe.net/db/support/db-terms-conditions.pdf

% Note: this output has been filtered.
%       To receive output for a database update, use the "-B" flag.

% Information related to 'AS12557 - AS13223'

as-block:      AS12557 - AS13223
descr:         RIPE NCC ASN block
remarks:       These AS Numbers are assigned to network operators in the RIPE NCC service region.
mnt-by:        RIPE-NCC-HM-MNT
created:       2018-11-22T15:27:24Z
last-modified: 2018-11-22T15:27:24Z
source:        RIPE

% Information related to 'AS12996'

% Abuse contact for 'AS12996' is 'abuse@domeneshop.no'

aut-num:       AS12996
as-name:       DOMENESHOP
descr:         Oslo, Norway
import:        from AS174 accept ANY
% Truncated text
org:           ORG-DA421-RIPE
admin-c:       SS19786-RIPE
tech-c:        HH2777-RIPE
status:        ASSIGNED
mnt-by:        RIPE-NCC-END-MNT
mnt-by:        DOMENESHOP-MNT
remarks:       -----
remarks:       For peering info, please see: https://peeringdb.com/net/5479
remarks:       -----
created:       2002-09-09T21:41:27Z
last-modified: 2021-11-29T10:57:03Z
source:        RIPE # Filtered

organisation:  ORG-DA421-RIPE
org-name:      Domeneshop AS
country:       NO
org-type:      LIR
address:       Christian Krohgs gate 16
address:       0186
address:       Oslo
address:       NORWAY
phone:         +4722943333
fax-no:        +4722943334
mnt-ref:       DOMENESHOP-MNT
mnt-ref:       RIPE-NCC-HM-MNT
mnt-by:        RIPE-NCC-HM-MNT
mnt-by:        DOMENESHOP-MNT
abuse-c:       HH2777-RIPE
created:       2012-06-07T11:55:20Z
last-modified: 2020-12-16T13:26:59Z
source:        RIPE # Filtered
admin-c:       SS19786-RIPE
```

```

role:          Domeneshop Hostmaster
address:       Domeneshop AS
address:       Christian Krohgs gate 16
address:       0186 Oslo
address:       Norway
phone:         +47 22 94 33 33
fax-no:        +47 22 94 33 34
abuse-mailbox: abuse@domeneshop.no
admin-c:       SS19786-RIPE
tech-c:        SS19786-RIPE
nic-hdl:       HH2777-RIPE
mnt-by:        DOMENESHOP-MNT
created:       1970-01-01T00:00:00Z
last-modified: 2021-03-06T09:54:48Z
source:        RIPE # Filtered

```

```

person:        Stale Schumacher
address:       Domeneshop AS
address:       Christian Krohgs gate 16
address:       0186 Oslo
address:       Norway
phone:         +47 22 94 33 33
fax-no:        +47 22 94 33 34
nic-hdl:       SS19786-RIPE
mnt-by:        DOMENESHOP-MNT
created:       2012-06-07T12:47:02Z
last-modified: 2021-03-06T09:53:34Z
source:        RIPE # Filtered

```

% This query was served by the RIPE Database Query Service version 1.104 (BLAARKOP)

You can also use `whois` on a domain, although in recent years this information has become more limited. Compare the query for `noroff.no` with that for `cnn.com`. The output below has been abridged for readability. You should run these commands and look at the full output.

```
$ whois noroff.no
```

```
% Truncated text
Domain Information
```

```

NORID Handle.....: NOR30169D-NORID
Domain Name.....: noroff.no
Registrar Handle.....: REG42-NORID
Tech-c Handle.....: DH38R-NORID
Name Server Handle.....: NSHY11H-NORID
Name Server Handle.....: NSHY46H-NORID
Name Server Handle.....: NSHY81H-NORID
DNSSEC.....: Signed
DS Key Tag      1.....: 19061
Algorithm       1.....: 13
Digest Type     1.....: 2
Digest          1.....: e923453b161a15d3685a00e76ff1807b014d78cc916b4f12d8092aea1b93a814
DS Key Tag      2.....: 19061
Algorithm       2.....: 13
Digest Type     2.....: 4
Digest          2.....: caeed516515e1b8a933e3ba72dae984f13bc602e7a45b3a10bbcc609b876e2c144fd4feb25

```

```
Additional information:
```


Created: 2015-07-03
Last updated: 2022-10-19

```
$ whois cnn.com
Domain Name: CNN.COM
Registry Domain ID: 3269879_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.corporatedomains.com
Registrar URL: http://cscdbs.com
Updated Date: 2018-04-10T16:43:38Z
Creation Date: 1993-09-22T04:00:00Z
Registry Expiry Date: 2026-09-21T04:00:00Z
Registrar: CSC Corporate Domains, Inc.
Registrar IANA ID: 299
Registrar Abuse Contact Email: domainabuse@cscglobal.com
Registrar Abuse Contact Phone: 8887802723
Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
Domain Status: serverDeleteProhibited https://icann.org/epp#serverDeleteProhibited
Domain Status: serverTransferProhibited https://icann.org/epp#serverTransferProhibited
Domain Status: serverUpdateProhibited https://icann.org/epp#serverUpdateProhibited
Name Server: NS-1086.AWSDNS-07.ORG
Name Server: NS-1630.AWSDNS-11.CO.UK
Name Server: NS-47.AWSDNS-05.COM
Name Server: NS-576.AWSDNS-08.NET
DNSSEC: unsigned
URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/
>>> Last update of whois database: 2022-10-21T07:25:07Z <<<
```

% Truncated text

```
Domain Name: cnn.com
Registry Domain ID: 3269879_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.corporatedomains.com
Registrar URL: www.cscprotectsbrands.com
Updated Date: 2020-10-20T13:09:44Z
Creation Date: 1993-09-22T00:00:00.000-04:00
Registrar Registration Expiration Date: 2026-09-21T00:00:00.000-04:00
Registrar: CSC CORPORATE DOMAINS, INC.
Registrar IANA ID: 299
Registrar Abuse Contact Email: domainabuse@cscglobal.com
Registrar Abuse Contact Phone: +1.8887802723
Domain Status: clientTransferProhibited http://www.icann.org/epp#clientTransferProhibited
Domain Status: serverDeleteProhibited http://www.icann.org/epp#serverDeleteProhibited
Domain Status: serverTransferProhibited http://www.icann.org/epp#serverTransferProhibited
Domain Status: serverUpdateProhibited http://www.icann.org/epp#serverUpdateProhibited
Registry Registrant ID:
Registrant Name: Domain Name Manager
Registrant Organization: Turner Broadcasting System, Inc.
Registrant Street: One CNN Center
Registrant City: Atlanta
Registrant State/Province: GA
Registrant Postal Code: 30303
Registrant Country: US
Registrant Phone: +1.4048275000
Registrant Phone Ext:
Registrant Fax: +1.4048271995
Registrant Fax Ext:
Registrant Email: tmggroup@turner.com
Registry Admin ID:
Admin Name: Domain Name Manager
```

Admin Organization: Turner Broadcasting System, Inc.
Admin Street: One CNN Center
Admin City: Atlanta
Admin State/Province: GA
Admin Postal Code: 30303
Admin Country: US
Admin Phone: +1.4048275000
Admin Phone Ext:
Admin Fax: +1.4048271995
Admin Fax Ext:
Admin Email: tmgroup@turner.com
Registry Tech ID:
Tech Name: TBS Server Operations
Tech Organization: Turner Broadcasting System, Inc.
Tech Street: One CNN Center
Tech City: Atlanta
Tech State/Province: GA
Tech Postal Code: 30303
Tech Country: US
Tech Phone: +1.4048275000
Tech Phone Ext:
Tech Fax: +1.4048271593
Tech Fax Ext:
Tech Email: hostmaster@turner.com
Name Server: ns-1086.awsdns-07.org
Name Server: ns-1630.awsdns-11.co.uk
Name Server: ns-47.awsdns-05.com
Name Server: ns-576.awsdns-08.net
DNSSEC: unsigned

% Truncated Text

ASN queries with Whois

Whois can be used to query a number of online databases for information about network objects. One of the other useful things you can do is use it to query what netblocks are considered to be related to a given Autonomous system number (ASN). For this we will use the Routing Advertisement Database ([RADB](#)) service. This service is built up based on the BGP routing tables used on the global Internet, and is intended to provide a 'current' view.

Taking the AS value from our earlier look up for 194.63.248.52 in the Team Cymru service we see that is in AS12996 (the AS being the first column returned). We know that this AS belongs to DOMENESHOP, but may be interested in knowing what other network blocks are associated with this AS. The rather involved command below returns this:

```
$ whois -h whois.radb.net -- '-i origin AS12996' | grep -Eo "([0-9.]{4})/[0-9]+" | head
151.249.124.0/24
151.249.125.0/24
194.63.248.0/21
194.63.253.0/24
194.63.248.0/23
194.63.250.0/24
194.63.252.0/24
194.63.254.0/24
194.63.251.0/24
185.134.244.0/22
...
```

The AS you wish to query, should come after the `-i origin`. The Regular expression strips out only the information we are interested in. The raw data returned, contains additional information:

```
$ whois -h whois.radb.net -- '-i origin AS12996' | head
route:      151.249.124.0/24
origin:     AS12996
descr:      Webair BGP Customer Space
mnt-by:     MAINT-AS27257
changed:    sagi@webair.com 20190927 #19:39:27Z
source:     RADB
```

...

While whois is useful as a general tool, there are specialist services and some excellent Python packages, that make it much easier to use in your scripts, than having to process the raw output. More information on advanced usage can be found in ARIN's [online guide](#).

Working on Windows

If you are not familiar with these tools, or feel you need a refresh, work through the following brief tutorial on WHOIS, NSlookup, Host and Dig:

The following guide shows how this can be achieved in Windows. You should perform the same tasks on your Linux system (taking note that you should ignore the `.exe` extensions of course!). Alternately you can use Windows, but may find it more difficult to integrate the output of commands with your other data. <https://securityonline.info/tutorial-nslookuphostdigwhois-dns-information-gathering/>

Application

Having worked through the examples above, use your newfound skills to attempt to answer the following as a way of exploring:



1. Does Noroff.no use DNSSEC ?
2. What are the mail exchanger(s) for `nrk.no`?
3. What are the authorized mailservers for `investec.com` ?
4. When was `nuc.no` registered ?
5. Who is the domain registrar `bbc.com` ?
6. Who is the abuse contact for `146.231.0.0/16` ?



You should **very much avoid** using the many web based tools that are available to solve these tasks. In almost all cases, these are often near impossible to script, or may not work if you have to run queries for a certain IP range. They **do** however serve as a useful way to check/confirm your results. **You need to be familiar with command line tools**. The biggest **downfall** of web based tools is that they are not scriptable, making handling volumes of data both difficult and error prone.



You are strongly encouraged to try looking up information on other domains, IP addresses and netblocks. This will help make sure you understand what you are doing, and ensure you are comfortable using the tools.

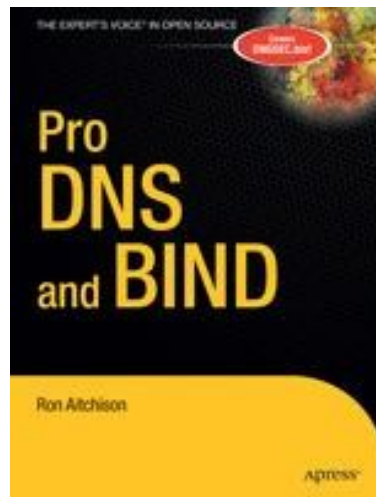


ASN to IP

The `whois` command line is not always the easiest to work with. Fortunately there are a number of libraries which can make it easier when you are working with data programatically. `asn-to-ip.py` is a tool that queries the RADB whois and returns IP netblocks associated with BGP Autonomous System Numbers (ASN). Returns IPv4 results by default, adds IPv6 results as an option. Can operate either on the command line or as a basic web service, and a Docker container is available. <https://github.com/ddimick/asn-to-ip>



DNS provides a wealth of information, but relatively few people have a solid grasp on how it operates. *Pro DNS and BIND* by Ron Aitchison, is an excellent reference for anyone working in DNS operations, or as a desk reference as to the some more esoteric means and modes of the DNS system and DNS server (especially BIND) operation. It also provides a great introduction to DNSSEC.



3 Packets Revisited

Building on work you did last week, and in the preceding two tasks this week explore the information contained in this the data for this Task. Download the file **Tut7.zip** from Moodle. The password for this is the same as for this is provided in Lectures. The cryptographic checksums can be found in the file `checksum.txt` on Moodle. You should validate your download is correct before proceeding.

This file contains raw packet data (pcap format) in the file `sensor_201910.cap` (236MB uncompressed). This data was recorded from the same sensor system as the examples you dealt with last week. The traffic is one way (there are no responses). There is no filtering in front of this system, so the traffic received is typical of what can be seen on internet facing hosts.

*For the purposes of **this** tutorial task, ICMP has been **removed** from the data.* This is due to the issues `tshark` has when processing this data and a number of people still not quite understanding what needs to be done to deal with it. The data was collected and processed on a system that was set to **GMT+2**.



Notes

Much of what you do when working with network data is repetitive. You should keep careful notes of snippets of command line or useful set of filters that may be of use in the future.

The primary purpose of this task is to provide you with an opportunity to further practice your skills working with network data. Understanding and being able to process, analyse and manipulate network data are important skillsets relating to network defence. You are strongly encouraged to use this dataset as a basis for further exploration.



Timezones

When working with network traffic, or other data from remote systems, you should be aware of what timezone the system was using when capturing data. In some cases this will be specified, but often you may need to infer based on other factors. Having a common view of the time is critical when trying to correlate logs and events across systems. You ideally should set the timezone of your analysis system to the same as that of your data.

A good indication there may be a timezone problem is when your events are off by hours, or a dataset for a month is off by a day on one side or another.

Setting your Timezone

The timezone of a system is typically set at install time. You can check the timezone on your system using the following commands.

```
$ date
Mon Oct 24 09:41:51 AM UTC 2022
$ cat /etc/timezone
Etc/UTC
```

This system is set to UTC (also known as GMT, Zulu, Zero, or base time). A more detailed output of the time configuration on your Ubuntu system can be found using `timedatectl`.

```
$ timedatectl
    Local time: Mon 2022-10-24 09:46:42 UTC
    Universal time: Mon 2022-10-24 09:46:42 UTC
        RTC time: Mon 2022-10-24 09:44:26
        Time zone: Etc/UTC (UTC, +0000)
System clock synchronized: yes
      NTP service: active
    RTC in local TZ: no
```

This information also shows that the NTP service is active. This is a good way of ensuring your system has a consistent view of time and is generally accurate to within a few milliseconds. You can use `timedatectl` to list available time zones that you can then use to set your system.

```
$ timedatectl list-timezones
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
....
```

You can use `grep` to check for particular timezones eg in Europe. If you specifically want a GMT offset you should look at the "Etc/" timezones that are listed.

```
$ timedatectl list-timezones | grep Etc
Etc/GMT
Etc/GMT+0
Etc/GMT+1
Etc/GMT+2
....
Etc/UCT
Etc/UTC
Etc/Universal
Etc/Zulu
```

Given this information you can use `timedatectl set-timezone` to change the machines timezone.

```
$ sudo timedatectl set-timezone Europe/Oslo
```



What Timezone? Working out what timezones relate to what offsets from UTC is beyond the scope of the tutorial. There are many online sites that can help you with this. For the purposes of this course the important timezones are GMT/UTC +0000 ; SAST +0200 ; and CEST +0200. Note that CEST is **not** the same as CET which does not have the addition hour for daylight saving. South African Standard Time (SAST) does not change during the year and remains at UTC+2 and is available in the timezone Africa/Johannesburg

Re running the `timedatectl` we can see that the system timezone has changed, and that the time offset from UTC/GMT is +0200 or two hours.

```
$ timedatectl
      Local time: Mon 2022-10-24 11:53:46 CEST
      Universal time: Mon 2022-10-24 09:53:46 UTC
          RTC time: Mon 2022-10-24 09:53:47
          Time zone: Europe/Oslo (CEST, +0200)
System clock synchronized: yes
          NTP service: active
      RTC in local TZ: no
```



Alternate methods

An alternate method for changing the timezone that uses the console base prompting system is to prompt the re-configuration of the `tzdata` package. This can be done using:

```
sudo dpkg-reconfigure tzdata
```

Follow the prompts.



Exploring Data

Some guiding questions you can use to explore your data are given below. Ensure you keep notes of how you process the data so you are able to undertake similar processing with other datasets. Read the **entire task** before starting to explore the data.

1. Isolate any spoofed traffic originating from the RFC1918 Address blocks, and write this to a separate file: `rfc1918.cap`
 - (a) How many distinct source IP Addresses were observed ?
 - (b) By comparing the TTL value of traffic one can potentially differentiate between sources of spoofed addresses. For the RFC1918 addresses in your isolated file, how many different TTL values were observed for each. Which source address in this capture file had the most TTL values?
2. When did the main capture (`sensor_201910.cap`) start and stop ?
3. What was the network being used to capture this traffic? How big is it relative to what you saw previously?
4. For each of TCP and UDP traffic:
 - (a) How many distinct source IP Addresses were observed on this protocol?
 - (b) What proportion of the total traffic was this protocol? Calculate based on the number of hosts and the number of packets.
 - (c) How many hosts sent **both** TCP and UDP traffic.
5. Isolate a single 24 hour period of traffic from `sensor_201910.cap` starting on the 15th at midnight (00:00:00) GMT+2, and then answer the following:
 - (a) How many packets were captured during this period?
 - (b) How many unique sources were seen for each of UDP and TCP?
 - (c) What were the top 10 destination ports based on the number of packets for TCP
 - (d) How does this change if you rank the ports by the number of hosts targeting them
 - (e) What are the Top 10 Countries (based on the number of source hosts)?
 - (f) How many distinct netblocks did the top ranked AS have (ranked by the number of distinct IP addresses observed)?
 - (g) What percentage of the attributable netblocks to this ASN did you observe?
 - (h) What number and percentage of the packets are likely benign based on the information in `shodan.txt` used last week.

This dataset will be revisited in the next tutorial

**Domain specific engines for CTI**

There are a number of search engines which hold, or can be used to find security specific data of the type commonly used for CTI. One needs to use these systems directly, and they often provide a lot more relevant information than on the public web as indexed by Google and others.

- Shodan- <https://www.shodan.io/>
- Censys.io - <https://censys.io/>
- Abuse IP Database - <https://www.abuseipdb.com/>
- Robtex Domain/IP Lookup - <http://itools.com/tool/robtex-domain-ip-address-lookup>
- NetCraft - <http://itools.com/tool/robtex-domain-ip-address-lookup>
- GitHub Code Search - <https://github.com/search>
- Yandex Image Translate - <https://translate.yandex.com/ocr>
- Binary Edge - <https://www.binaryedge.io/>
- Threat Connect - <https://threatconnect.com/free/>
- Threat Crowd - by OTX -<https://www.threatcrowd.org/>

4 Linux Network Stack Tuning

The Linux network stack is a very mature and flexible implementation. While the primary protocol suite used on most system is TCP/IP, a number of other protocols are supported. This task is focused on the TCP/IP related settings that can be tuned in the stack. Some of these are security feature, others are to enable more rarely used features that maybe required in some situations.



This task is to be done on the Ubuntu 22.04 Server you have been setting up during the course.

Basic Hardening

Some of the basic areas to consider when hardening relate to risk present from abuses of the TCP/IP protocol functionality.



Work through the tasks below setting these tunables via the command line and comparing them to default values. For each consider when it would be appropriate to use and how it adds to the overall security of the system.

tcp_syncookies

This parameter can help to prevent SYN flood DDoS attacks by testing the validity of the SYN packets. For security reasons it is recommended to enable the parameter. Note that the process is conducted without consuming memory or connection resources.

```
net.ipv4.tcp_syncookies = 1
```

ignore_broadcasts

A ping broadcast is used to see what hosts in LAN are up, but there are other ways to do this. It is safer to disable this option because while these can constitute legitimate traffic in many circumstances, unless they are specifically needed, ICMP (ping) broadcasts and multicasts are usually a sign of Smurf attack.

```
net.ipv4.icmp_echo_ignore_broadcasts = 1
```

accept_redirects

The parameter allows to enable or disable ICMP redirects acceptance. ICMP redirects are important to routers, but can create security problems for servers, so it is recommended to set the parameter to off.

```
net.ipv4.conf.all.accept_redirects = 0
```

accept_source_route

Tells *Netfilter* to accept or decline source routed packets. Source routed packets are security risk, because they can allow routing packets through an untrusted or insecure interface.

```
net.ipv4.conf.all.accept_source_route = 0
```

rp_filter

This parameter controls reverse path filtering, which tries to ensure packets use legitimate source addresses. When is turned on it can prevent some IP spoofing attacks.

```
net.ipv4.conf.all.rp_filter = 1  
net.ipv4.conf.default.rp_filter = 1
```

log_martians

The parameter allows to keep track of packets which could potentially indicate an attack on server. This packets are those that includes impossible IP addresses, bad source routing, bad redirect packets and others. Martians and Bogons are packets with specific addresses that are unexpected or should not be observed. you can read more about them [here](#).

```
net.ipv4.conf.all.log_martians = 1
```



Can you determine which network blocks this would include ?

send_redirects

Enables or disables ICMP redirects which are used mainly by routers to send out ICMP redirects to other hosts. For security reasons, it is recommended to disable this option.

```
net.ipv4.conf.all.send_redirects = 0
```

fin_timeout

Tells *Netfilter* how much seconds keep sockets in FIN-WAIT-2 state which means that connection is closed, and the socket is waiting for a shutdown from the remote end. Decreasing the value to 30 can avoid some DDoS attacks or other problems that arose from getting huge amounts of connections

```
net.ipv4.tcp_fin_timeout = 30
```

tcp_keepalive_time

Tells the *Netfilter* how often to send TCP keep-alive packets to keep an connection alive if it is currently unused. The value 1800, or 30 minutes, is a good value for most servers.

```
net.ipv4.tcp_keepalive_time = 1800
```

Persistence

You can persist values you want to be set at boot time by placing them in `/etc/sysctl.conf`. Take note of the format of entries used in this file.



What do you think the value is in explicitly setting values are in this file, where its the same as the default for a given kernel or particular Linux distribution?

Extending

The items above barely scratch the surface in terms of the options. Section B on page 32 contains an example of an extract from a `/etc/sysctl.conf` that relates to network security settings.



Using appropriate online tools and the [Linux Kernel Documentation](#) as a guide, ensure you have a good understanding of the following concepts and the tunables that relate to them.

- SYN attacks
- Packet forwarding
- IP spoofing protection
- Source route verification
- Redirects
- Source Routing
- Address Resolution Protocol
- Broadcasts



You can find more on some of these resources at the following sites:

- <https://www.cyberciti.biz/faq/linux-kernel-etcsysctl-conf-security-hardening/>
- <https://www.kmotoko.com/articles/linux-hardening-kernel-parameters-with-sysctl/>

5 Linux Logging

Most of the system information is stored in the system log `/var/log/syslog` by default. Information contained within this log may not be found in other logs but might be very useful in certain situations. Events from some applications which do not have their own log file, might be stored in the system log file which contains (for Debian derived distributions such as Ubuntu and Mint) also all logs that used to be written in the file `/var/log/messages`.

Authorisation related events are tracked in the authorisation log. This includes mechanisms like the PAM framework, `sshd` remote logins and the `sudo` command. The authentication log file, which is useful for user logins investigation as well as the `sudo` command usage, can be found under the directory `/var/log/auth.log`. To see `sshd` logins related information from the Authorisation Log file, you could use the `grep` based command below:

```
grep sshd /var/log/auth.log | less
```

You can get a lot more creative (and accurate) when using regular expressions with `grep` and `egrep` in conjunction with other command line work horse tools such as `sed` and `awk`.



Getting the Most out of `grep`

- [Using Grep & Regular Expressions to Search for Text Patterns in Linux](#)
- [Regular expressions in grep \(regex with examples\)](#)
- [Regular Expression in grep](#)



If you still trying to wrap your head around logging these [logging best practices](#) may be of come value to recap what was covered in lectures. As with many security related issue, OWASP has some good [advice on logging](#). While a longer read, this [web article](#) provides a very comprehensive guide to Linux logging.



The Complete Guide to Log and Event Management

Dr. Anton Chuvakin has produced a [whitepaper](#) which is described as:

Everybody has logs and that means that everybody ultimately will have to deal with them—if only because many regulatory mandates prescribe that. In this guide, Dr. Anton Chuvakin will analyse the relationship between SIEM and log management, focusing not only on the technical differences and different uses for these technologies but also on architecting their joint deployments. In addition, he will provide recommendations for companies that have deployed log management or SIEM so they can plot their roadmap for enhancing, optimising and expanding their deployment. He will also recommend a roadmap for companies that have already deployed both of these technologies.



Journalctl

`journalctl` is a tool present in newer Linux versions that provides an interface to a newer logging system based on `systemd`. You may find the following of use if you wish to extend your skillset to look at this tool:

- [Journalctl cheat sheet](#) and introduction to some common tasks.
- [How To Centralize Logs With Journald on Ubuntu 20.04](#)
- [How to use the journalctl Linux command](#)

6 Auditing



More Detailed Resources

Two good whitepapers from the SANS Reading room are:

- [Using Linux Scripts to Monitor Security](#) by Harvey Newstrom
- [IDS: File Integrity Checking](#) by Lawrence Grim



Linux Audit CheatSheets

- [Linux Compromise Assessment Command Cheat Sheet](#) (updated)
- [Cybersecurity Ops with bash](#)
- [Linux Command Cheat Sheet](#)
- [Some great 1-line commands for working with files PDF](#)
- [Linux one-line commands](#)
While some of these are redhat specific (referencing rpm) others are useful
- [USB device artefacts checksheet](#)



This Security Audit script to gathers information about your Linux system which can also help you in the process of hardening. The script check 28 different elements of a system and is available at <https://github.com/sokdr/LinuxAudit>

Another option is a simple set of [scripts](#) which implement some General system , Disk, File System and Networking Checks. The Linux Baseline Security Analyser ([LBSA](#) was based on the similarly named Microsoft tool), and is a more monolithic approach. Two other articles around auditing with a specific focus on Ubuntu Servers are:

- [Auditing Your Ubuntu Servers](#)
- [Ubuntu: Auditing sudo commands and forwarding audit logs using syslog](#)



The *Cybersecurity Ops with bash* book by Paul Troncone & Carl Albing has an accompanying Github site, where you can grab many of the scripts and examples which you may find useful. You can clone the git repo from <https://github.com/cybersecurityops/cyber-ops-with-bash>



Windows Focussed Resources

Some resources for undertaking security audits on Microsoft Windows platforms can be found here:

- <https://www.malwarearchaeology.com/cheat-sheets>
- <https://github.com/MalwareArchaeology/ATTACK>
- [PowerShell Commands for Pentesters](#)
- [Security incident log review checklist](#)



How-To Make Linux System Auditing a Little Easier

Paul Santos has produced a short [paper on Linux auditing](#). This contains discussion of various programs and utilities that can be used to audit your Linux system and how to put them all together in one script to make daily system auditing a little easier. This was done as part of a graduate program at SANS.org. More papers on [Auditing and Assessment](#) are available in their reading room.

Root Kits and Malware

`rkhunter` (Rootkit Hunter) is a Unix-based tool that scans for rootkits, backdoors and possible local exploits. It does this by comparing SHA-1 hashes of important files with known good ones in online databases, searching for default directories (of rootkits), wrong permissions, hidden files, suspicious strings in kernel modules, and special tests for Linux and FreeBSD.

This is the kind of thing you would probably want to run on a daily basis.



Install `rkhunter` using `apt`.

On Debian based system, `rkhunter` comes with cron scripts. To enable them check `/etc/default/rkhunter` or use `dpkg-reconfigure` and say **Yes** to all of the questions:

```
sudo dpkg-reconfigure rkhunter
```



You will very likely run into issues doing this due to problems with web updates. If you do, try the following changes² in your `/etc/rkhunter.conf`:

```
MIRRORS_MODE=1 ---> MIRRORS_MODE=0 line 122 - use any mirror
```

```
UPDATE_MIRRORS=0 ---> UPDATE_MIRRORS=1 line 107 - update mirror list
```

```
WEB_CMD="/bin/false" ---> WEB_CMD="" line 1189 - allow this all to work.
```

For further details see the `rkhunter` documentation.



After you've finished with all of the changes, make sure all the settings are valid:

```
sudo rkhunter -C
```

Update `rkhunter` and its database:

```
sudo rkhunter --versioncheck
```

```
sudo rkhunter --update
```

```
sudo rkhunter --propupd
```

If you want to do a manual scan and see the output:

```
sudo rkhunter --check
```



Additional Resources

- <http://www.chkrootkit.org/>
- <https://www.cyberciti.biz/faq/howto-check-linux-rootkist-with-detectors-software/>



Daily Reporting

How would you run this script and have it mail you the output every day?

²[Stack overflow to the rescue](#) rather than reading docs :)



Cron Format

A reminder of the entry format in the system cron configuration which is found in `\etc\crontab`.

```
#minute (0-59)
#|   hour (0-23)
#|   |   day of the month (1-31)
#|   |   |   month of the year (1-12 or Jan-Dec)
#|   |   |   |   day of the week (0-6 with 0=Sun or Sun-Sat)
#|   |   |   |   |   commands
#|   |   |   |   |   |
#### file integrity check
00 */6 * * * /yourscript parameters
# This runs every 6 hours as denoted by */6
```

chkrootkit

`chkrootkit` (Check Rootkit) is a common Unix-based program intended to help system administrators check their system for known rootkits. It is a shell script using common UNIX/Linux tools like the `strings` and `grep` commands to search core system programs for signatures and for comparing a traversal of the `/proc` filesystem with the output of the `ps` (process status) command to look for discrepancies.

Install the software

```
sudo apt install chkrootkit
```

And run a scan: `sudo chkrootkit`. Results should look similar to those below.

```
ROOTDIR is '/'
Checking 'amd'...          not found
Checking 'basename'...    not infected
Checking 'biff'...        not found
Checking 'chfn'...        not infected
Checking 'chsh'...        not infected
...
Checking 'scalper'...     not infected
Checking 'slapper'...     not infected
Checking 'z2'...         chklastlog: nothing deleted
Checking 'chkutmp'...     chkutmp: nothing deleted
Checking 'OSX_RSPLUG'...  not infected
```

Make a backup of `chkrootkit`'s configuration file `/etc/chkrootkit.conf`:

```
sudo cp --archive /etc/chkrootkit.conf /etc/chkrootkit.conf-COPY-$(date +"%Y%m%d%H%M%S")
```



1. Why do you think its a good idea to make copies of important files?
2. What is the resulting filename form the copy operation above?
3. Was it what you expected?
4. What advantage do you think there is to naming them with a scheme as above ?

Similar to `rkhunter` you would want `chkrootkit` to run every day and e-mail you the result. On Debian based system, `chkrootkit` comes with cron scripts. To enable them check `/etc/chkrootkit.conf` or use `dpkg-reconfigure` and say **Yes** to the **first** question:

```
sudo dpkg-reconfigure chkrootkit
```

Logwatch

logwatch is a system log analyzer and reporter. Your server will be generating a lot of logs that may contain important information. Unless you plan on checking your server everyday, you'll want a way to get e-mail summary of your server's logs. To accomplish this we'll use logwatch.

logwatch scans system log files and summarises them. You can run it directly from the command line or schedule it to run on a recurring schedule. logwatch uses service files to know how to read/summarize a log file. You can see all of the stock service files in `/usr/share/logwatch/scripts/services`.

logwatch's configuration file `/usr/share/logwatch/default.conf/logwatch.conf` specifies default options. You can override them via command line arguments.

Install logwatch using `apt` and then run it directly to see a sample of what it collects:

```
sudo /usr/sbin/logwatch --output stdout --format text --range yesterday --service all

##### Logwatch 7.4.3 (03/08/20) #####
Processing Initiated: Mon Aug  3 03:29:18 2020
Date Range Processed: yesterday
                      ( 2020-Aug-02 )
                      Period is day.
Detail Level of Output: 5
Type of Output/Format: stdout / text
Logfiles for Host: host
#####

----- Cron Begin -----
...
...
----- Disk Space End -----

##### Logwatch End #####
```



Go through logwatch's self-documented configuration file `/usr/share/logwatch/default.conf/logwatch.conf` before continuing. There is no need to change anything here but pay special attention to the Output, Format, MailTo, Range, and Service as those are the ones we'll be using.

For our purposes, instead of specifying our options in the configuration file, we will pass them as command line arguments in the daily `cron` job that executes logwatch. That way, if the configuration file is ever modified (e.g. during an update), our options will still be there.

Make a backup of logwatch's daily cron file `/etc/cron.daily/00logwatch` and unset the execute bit:

```
sudo cp --archive /etc/cron.daily/00logwatch /etc/cron.daily/00logwatch-COPY-$(date +"%Y%m%d%H%M%S")
sudo chmod -x /etc/cron.daily/00logwatch*
```

By default, logwatch outputs to `stdout`. Since the goal is to get a daily e-mail, we need to change the output type that logwatch uses to send e-mail instead. We could do this through the configuration file above, but that would apply to every time it is run – even when we run it manually and want to see the output to the screen.

Instead, we'll change the cron job that executes logwatch to send e-mail. This way, when run manually, we'll still get output to stdout and when run by cron, it'll send an e-mail. We'll also make sure it checks for all services, and change the output format to html so it's easier to read regardless of what the configuration file says. In the file `/etc/cron.daily/00logwatch` find the execute line and change it to:

```
/usr/sbin/logwatch --output mail --format html --mailto root --range yesterday --service all
```

You can test the cron job by executing it:

```
sudo /etc/cron.daily/00logwatch
```

Auditd

The Linux Auditing System is a native feature to the Linux kernel (first introduced in the 2.6.x series) that collects certain types of system activity to facilitate incident investigation. In this post, we will cover what it is as well as how people deploy and manage it. While a useful tool, it has its weaknesses, including excessive overhead, lack of granularity, missing container support, onerous output, and 'buginess'.

Use the following resources to guide you through the installation and testing of auditd.

- [What is the Linux Auditing System \(aka AuditD\)](#)
- [Auditd - Tool for Security Auditing on Linux Server](#)
- [Linux audit files to see who made changes to a file](#)
- [Auditd Linux Tutorial](#)

Lynis

Lynis is a battle-tested security tool for systems running Linux, macOS, or Unix-based operating system. It performs an extensive health scan of your systems to support system hardening and compliance testing. The project is open source software with the GPL license and available since 2007.



This is suggested if you would like to know more about system auditing.
[How to Perform Security Audits With Lynis](#)



Want to know more?

A paper entitled [IDS for logs: Towards implementing a streaming Sigma rule engine](#) by Markus Kont and Mauno Pihelgas was published at the 2020 NATO CCDCOE Cyber Warfare conference.

7 Deliverable

While the **entire** tutorial contents are important the following specific items are in scope for forthcoming engagement activities.

Enrichment

You required to have answered

- Task 1 Questions (a)-(e) (page 5)
- Task 3 Questions 1 - 5 (page 19)

These are in scope for the Engagement 3 Activity

You should be comfortable using the `whois`, `dig` and `nslookup` tools to find information at the command line. Similarly you should be comfortable using tools like `tshark` and `tcpdump` to filter and select specific traffic from a packet capture file. Task 4 provides some example questions for you to try.

A News

The following are new articles of interest in the last week. These are all relatively short, and you should get in the habit of staying aware of what is going on in the industry at large. The topics discussed here fall in scope for the engagement tasks.



News Week of 9 October 2023

BREACH [Hackers Steal Database of Users from the European Telecommunications Standards Institute](#)

hackers hit the European Telecommunications Standards Institute (ETSI) portal and likely stole a database containing a list of their online users. Not all attacks target a company's or institution's infrastructure only to deploy ransomware, even if that might always be the case if we pay attention to all the stories popping up in the media. Sometimes, hackers will simply try to exfiltrate information, as was the situation with the ETSI incident.

ATTACK [Typosquatting campaign delivers r77 rootkit via npm](#)

researchers have identified a new, malicious supply chain attack affecting the npm platform. The "typosquatting" campaign first appeared in August and pushed a malicious package, node-hide-console-windows, which downloaded a Discord bot that facilitated the planting of an open source rootkit.

VULNS [Looney Tunables: New Linux Flaw Enables Privilege Escalation on Major Distributions](#)

A new Linux security vulnerability dubbed Looney Tunables has been discovered in the GNU C library's ld.so dynamic loader that, if successfully exploited, could lead to a local privilege escalation and allow a threat actor to gain root privileges. The GNU C library, also called glibc, is a core library in Linux-based systems that offers foundational features such as open, read, write, malloc, printf, getaddrinfo, dlopen, pthread_create, crypt, login, and exit..

POLICY [Red Cross Publishes Rules of Engagement for Hacktivists During War](#)

The International Committee of the Red Cross (ICRC) is urging hacking groups involved in conflict during war to abide by a set of rules meant to protect the general population. According to the organization, which oversees and monitors the rules of war, an increasing number of civilian hackers are getting involved in military conflicts by means of digital operations, especially in the context of the Russian war in Ukraine..

RESEARCH [Russia mistakenly doxxes its own secret bases and spies](#)

A relatively obscure website of the Moscow City Hall has given away a list of "special consumers" on the Russian electricity grid. It includes facilities maintained by the country's military and security agencies. "Special consumers" are obviously locations that are too important to ever be disconnected from the grid – a constant flow of electricity should be available at all times, even in the event of blackouts or power shortages.

B Network Stack sysctl

```
###
### NETWORK SECURITY ###
###

# Prevent SYN attack, enable SYNcookies (they will kick-in when the max_syn_backlog reached)
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_syn_retries = 2
net.ipv4.tcp_synack_retries = 2
net.ipv4.tcp_max_syn_backlog = 4096

# Disable packet forwarding
net.ipv4.ip_forward = 0
net.ipv4.conf.all.forwarding = 0
net.ipv4.conf.default.forwarding = 0
net.ipv6.conf.all.forwarding = 0
net.ipv6.conf.default.forwarding = 0

# Enable IP spoofing protection
# Turn on source route verification
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1

# Disable Redirect Acceptance
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.conf.default.secure_redirects = 0
net.ipv6.conf.all.accept_redirects = 0
net.ipv6.conf.default.accept_redirects = 0

# Disable Redirect Sending
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0

# Disable IP source routing
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0
net.ipv6.conf.all.accept_source_route = 0
net.ipv6.conf.default.accept_source_route = 0

# Don't relay bootp
net.ipv4.conf.all.bootp_relay = 0

# Disable proxy ARP
net.ipv4.conf.all.proxy_arp = 0
net.ipv4.conf.all.arp_ignore = 1
net.ipv4.conf.all.arp_announce = 2

# Mitigate time-wait assassination hazards in TCP
net.ipv4.tcp_rfc1337 = 1

# Enable bad error message Protection
net.ipv4.icmp_ignore_bogus_error_responses = 1

# Enable ignoring broadcasts request
net.ipv4.icmp_echo_ignore_broadcasts = 1
```

```
# Ensure that subsequent connections use the new values
# PUT TO THE END
net.ipv4.route.flush = 1
net.ipv6.route.flush = 1
```

System Tools and Files

Computer Network Defence - 2023/24

Version 1.01 20/10/2023

This weeks Tutorial concentrates on skills development and instruction rather than problem solving and application of knowledge. While continuing with the auditing theme explored last week, the tasks this week provide a guided exploration of some tools and techniques commonly used on unix systems. You may find useful in your upcoming assessment.

While a number of examples are provided, you are encouraged to use these as a basis for the development of further experimentation to help you learn and solidify your understanding. You should make use of this week to take the opportunity to consolidate material thus far.

In addition to these guided demonstrations, you should also be spending time this week working on the prepared examination question. In addition you should be using the time to catch up and ensure you are up to date.

If anyone is looking for further extension/practice activities, please get in contact, and I'll provide some guidance.

The practical tasks this week consist of three distinct groupings:

LOGGING *[Task 1]* The first task refreshes some of the logging components and concepts previously seen

AUDIT *[Tasks 2 & 3]* System auditing both for integrity as well as looking for signs of certain actions is an important part of overall network security. Task 2 explores how to go about locating and finding information on Linux system such as we use. This can be useful in a variety of cases, including Backup and Audit. Finally Task 3 look at some methods that can be used for helping determining system integrity. These two tasks are intended to provide an overview of some of the processes/skill needed in order to do this. These are not intended to be time consuming problem solving exercises.

READING There are five brief news articles to review, along with a reading and listening activity. These are now for **interest only**.



These broad areas can all be worked on independently, with no inter-dependency between the sections. The order in which you approach these three areas is up to you. You are strongly encouraged to switch between tasks if you get stuck or find your concentration drifting.

This tut *may* be updated during the course of the week with additional support material, make sure you have the latest version as shown on Moodle.

Contents

1	Linux Logging	1
2	Finding Files	2
3	System Packages	14
4	Deliverable	19
	Resources	19
A	News	20
B	Online Skill Builders	21

CHANGELOG

-  v1.0 - Initial Release
 -  v1.01 - Minor typos
-

1 Linux Logging

Most of the system information is stored in the system log `/var/log/syslog` by default. Information contained within this log may not be found in other logs but might be very useful in certain situations. Events from some applications which do not have their own log file, might be stored in the system log file which contains (for Debian derived distributions such as Ubuntu and Mint) also all logs that used to be written in the file `/var/log/messages`.

Authorisation related events are tracked in the authorisation log. This includes mechanisms like the PAM framework, `sshd` remote logins and the `sudo` command. The authentication log file, which is useful for user logins investigation as well as the `sudo` command usage, can be found under the directory `/var/log/auth.log`. To see `sshd` logins related information from the Authorisation Log file, you could use the `grep` based command below:

```
grep sshd /var/log/auth.log | less
```

You can get a lot more creative (and accurate) when using regular expressions with `grep` and `egrep` in conjunction with other command line work horse tools such as `sed` and `awk`.



Getting the Most out of `grep`

- [Using Grep & Regular Expressions to Search for Text Patterns in Linux](#)
- [Regular expressions in grep \(regex with examples\)](#)
- [Regular Expression in grep](#)



If you still trying to wrap your head around logging these [logging best practices](#) may be of come value to recap what was covered in lectures. As with many security related issue, OWASP has some good [advice on logging](#). While a longer read, this [web article](#) provides a very comprehensive guide to Linux logging.



The Complete Guide to Log and Event Management

Dr. Anton Chuvakin has produced a [whitepaper](#) which is described as:

Everybody has logs and that means that everybody ultimately will have to deal with them—if only because many regulatory mandates prescribe that. In this guide, Dr. Anton Chuvakin will analyse the relationship between SIEM and log management, focusing not only on the technical differences and different uses for these technologies but also on architecting their joint deployments. In addition, he will provide recommendations for companies that have deployed log management or SIEM so they can plot their roadmap for enhancing, optimising and expanding their deployment. He will also recommend a roadmap for companies that have already deployed both of these technologies.



Journalctl

`journalctl` is a tool present in newer Linux versions that provides an interface to a newer logging system based on `systemd`. You may find the following of use if you wish to extend your skillset to look at this tool:

- [Journalctl cheat sheet](#) and introduction to some common tasks.
- [How To Centralize Logs With Journald on Ubuntu 20.04](#)
- [How to use the journalctl Linux command](#)

2 Finding Files

Regular auditing of a system is important. Many of these audit functions are extensions of every-day systems administration activities. This task walk you through, and provide examples of the use of some of the more important functionality that is available using tools already built into the operating system.

These tools are useful for general navigation and use of a unix like system, such as the Ubuntu server in this course. They also perform a lot of the 'heavy lifting' that would otherwise make scripting quite difficult.

At the core of most tasks around exploring, auditing or even general operation on systems is working with files. Within this often the first challenge is trying to find files. This may be files of a certain type (such as images), size¹ (what is consuming disk space?), belonging to a given user, or those with certain permissions. There are many tools that can be used. The tools illustrated here are generally part of the bundled operating system, and can be used as a basis to build on when scripting or working at the command line.

The examples following demonstrate the most important (and commonly used) features of the tools, but are by no means complete. You are encouraged to see what other functionality they have by referring to their manual pages, or other online resources.

Using locate

The first of these is to use `locate`. This command relies on a database that is updated periodically, typically being run daily as part of the `cron` tasks on a system. On an Ubuntu system the script that does this is located in `/etc/cron.daily/plocate` and the actual database in `/var/lib/plocate/`. You can try `locate` using the following syntax, where *term* is what you are searching for: `locate term`

Running a search for "passwd" returns a long list of 187 entries on a standard such as we have been following (this may vary depending how files have been named on your system, or files you have created).

```
$ locate passwd
/etc/passwd
/etc/passwd-
/etc/pam.d/chpasswd
/etc/pam.d/passwd
/etc/security/opasswd
/snap/core20/1611/etc/passwd
...
/usr/share/man/man1/gpasswd.1.gz
/usr/share/man/man1/grub-mkpasswd-pbkdf2.1.gz
/usr/share/man/man1/openssl-passwd.1ssl.gz
/usr/share/man/man1/passwd.1.gz
/usr/share/man/man1/passwd.1ssl.gz
/usr/share/man/man5/passwd.5.gz
/usr/share/man/man8/chpasswd.8.gz
/usr/share/man/man8/chpasswd.8.gz
/usr/share/man/man8/update-passwd.8.gz
...
/var/lib/dpkg/info/passwd.list
/var/lib/dpkg/info/passwd.md5sums
/var/lib/dpkg/info/passwd.postinst
/var/lib/dpkg/info/passwd.postrm
/var/lib/dpkg/info/passwd.preinst
/var/lib/dpkg/info/passwd.prerm
```

This returns all files that contain "passwd" in their name or path. While useful it does not help find specifics, and one typically needs to do further filtering using tools such as `grep`. It is important to take note that the *term* is case sensitive. Have a look at the difference you see when running `locate man` and `locate Man`. To

¹You should already have come across and used the `du` command to help do this. There is a companion tool `df` which shows the disk space free.

perform a case insensitive search you can use `-i` or `--ignore-case` before the term. Regular expressions are supported directly by `locate`, but it is often faster and more semantically clear in scripts to use `grep`.

Returning to the example searching for files with names or paths containing "man", you can limit the search to just **filenames** rather than the word appearing in the path using the `-b` option.

```
$ $cnd@cnd22:~$ locate man | wc -l
5604
cnd@cnd22:~$ locate -b man | wc -l
948
$ locate -b man | tail -50
/var/lib/command-not-found/commands.db
/var/lib/command-not-found/commands.db.metadata
/var/lib/dpkg/info/command-not-found.conf
/var/lib/dpkg/info/command-not-found.list
/var/lib/dpkg/info/command-not-found.md5sums
/var/lib/dpkg/info/git-man.list
/var/lib/dpkg/info/git-man.md5sums
/var/lib/dpkg/info/libsemanage-common.conf
/var/lib/dpkg/info/libsemanage-common.list
...
```

Considering the output above we can see that reducing the search to only files containing the phrase "man" results in a substantial reduction. If you look at the actual output produced this is predominantly due to no longer returning of every manual (man) page on the system `locate` in `/usr/share/man/`.

Using `locate` can prove useful if you remember the name of a file, or a partial name, but not where on a system it is located. Be aware that the command can only report what is in its database. you can see when the database was last updated, by looking at the modification date of `/var/lib/plocate/plocate.db`.

```
$ ls -lad /var/lib/plocate/plocate.db
-rw-r----- 1 root plocate 2776299 Oct 27 05:22 /var/lib/plocate/plocate.db
```



Out of Date?

On 'normal' systems which are powered on constantly, the normal daily cron processes ensure that the database is updated. This is often not the case on a virtual machine such as being used in the course that is being powered on periodically. In order to force an update you can run `sudo updatedb`. Generally this should be avoided in production, but helps to move things along in a training environment.

You can try this by creating a new file in your home directory, and then testing for its existence with `locate`. It should not be returned. Force an update and try again, at which point you should see the file.

Using which

Another way of finding files is `which`. This command works slightly differently in terms of it does not have a database that it uses, but rather searches the system 'live'. The search is constrained to the currently set `PATH` environment variable. This variable tells your current shell (`bash`) where to look for commands when you instruct it to execute them. This is what allows you to type `ls` instead of having to fully specify it by a path such as `/usr/bin/ls`. The lack of your home directory being in the `PATH` is why you typically have you pre-fix your scripts with `./`. By doing so you explicitly instruct the shell to look for the file in the current directory.

We can use `which` to determine the full paths for commands as shown below:

```
$ which ssh
/usr/bin/ssh
$ which passwd
/usr/bin/passwd
$ which dpkg
/usr/bin/dpkg
$ which which
/usr/bin/which
```

This lets us know which command is being executed. Something to bear in mind is that by default `which` returns the first match it finds in the path. If you want all matches you need to use the `-a` flag.

```
$ which ls
/usr/bin/ls
$ which -a ls
/usr/bin/ls
/bin/ls
```

In this case `/usr/bin/ls` is returned because `/usr/bin` is listed before `/bin` in the `PATH` environment. You can see the current state of the path using `echo $PATH` for this only. Alternately you can use `env` to dump the entire environment.

```
cnd@cnd:~$ echo $PATH
/home/cnd/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:
/usr/local/games:/snap/bin
```

Using dpkg

Ubuntu uses a *package manager* to manage all the software components that make up the base operating system as well as any add-on application, tools and utilities that users may install. Keeping track of what file belongs to what package is nearly impossible for a person (the demo system we are using has over 120K files on it!). Fortunately a large part of the package management system is doing exactly this. If you want to know what package a specific file belongs to you can use `dpkg` or `dpkg-query`.

```
$ dpkg -S /usr/bin/who
coreutils: /usr/bin/who
$ dpkg-query -S /usr/bin/who
coreutils: /usr/bin/who
```

In both cases this returns that the `who`, as specified by its full path is part of the `coreutils` package. If you do not know the full path for a command its possible to combine this with `which` to handle that part for you. Make sure you use backticks ' rather than single quotes ' to tell the shell to execute this. Alternately you can use the `$()` syntax, which is becoming generally more preferred. Both are shown below.

```
$ dpkg -S `which sshd`
openssh-server: /usr/sbin/sshd
$ dpkg -S $(which sshd)
openssh-server: /usr/sbin/sshd
```

You can also pass an unqualified filename to these commands in which case they will return all files and packages that match.

```
/$ dpkg -S who
bash-completion: /usr/share/bash-completion/completions/ldapwhoami
coreutils: /usr/bin/who
coreutils: /usr/share/man/man1/whoami.1.gz
sosreport: /usr/lib/python3/dist-packages/sos/report/plugins/virtwho.py
coreutils: /usr/bin/whoami
coreutils: /usr/share/man/man1/who.1.gz
apport: /usr/share/apport/whoopsie-upload-all
byobu: /usr/lib/byobu/whoami
```

The `dpkg` tool has lots of other functionality that can be used to make your life easier these are explored in [section 3](#).

Using find

When looking for further information, and interrogating the state of a running system, the `find` command is one of the most powerful tools one can use at the command line. This tool can be used to do a lot of 'heavy lifting' relating to files, and file metadata.

Filenames

In its very simplest form `find` will iterate through the directory it is passed and print out directory and filenames. The output below shows this being run in a user home directory. *The actual files names Will vary on your system.*

```
$ cd ~
$ find .
.
./test.cap
./tlscheck.sh
./profile
./bashrc
./jaberwock
./viminfo
./bash_logout
./test3
./test2.cap
./gnupg
./gnupg/pubring.kbx~
./gnupg/pubring.kbx
./gnupg/trustdb.gpg
./gnupg/private-keys-v1.d
./gnupg/private-keys-v1.d/1E559E8E430FFE87BBEEFC4K363BEE8450B8383A176.key
./gnupg/private-keys-v1.d/82E9CE4AB4FAB84CAFEBABE5FAE8A904983CF2CDC.key
./gnupg/openpgp-revocs.d
./gnupg/openpgp-revocs.d/EFD62D9A07DEADBE3F51B42B3BD9C0EB19529D7C1F21D.rev
./gnupg/mykey.asc
./bash_history
./lessht
./test2b
./bin
./bin/ls
./bin/example.sh
./ssh
./ssh/authorized_keys
./test.pcap
./cache
./cache/motd.legal-displayed
./sudo_as_admin_successful
./testing
./testing/script.log
./testing/hostname.md5
$
```

While useful to generate listing for a system, this is not really useful for much else. Fortunately there are a number of arguments that can be passed to use the power and flexibility of the tool. The first of these is the `-name` option, and its corresponding `-iname` which ignored case. These are used to provide a pattern or 'glob' that `find` should use to filter its results. Here we are looking for files in `/var/log` starting with "a". Both the normal and case insensitive variants are shown. Multiple paths may be passed before options such as `-name`. Files on your system may vary.

```
$ find /var/log -name a*
/var/log/apparmor
/var/log/alternatives.log
/var/log/alternatives.log.1
/var/log/installer/autotinstall-user-data
/var/log/auth.log.2.gz
/var/log/auth.log.3.gz
/var/log/apt
/var/log/auth.log.1
/var/log/auth.log.4.gz
/var/log/alternatives.log.2.gz
/var/log/auth.log
```

```
$ find /var/log -iname A*
/var/log/apparmor
/var/log/alternatives.log
/var/log/alternatives.log.1
/var/log/installer/autotinstall-user-data
/var/log/auth.log.2.gz
/var/log/auth.log.3.gz
/var/log/apt
/var/log/auth.log.1
/var/log/auth.log.4.gz
/var/log/alternatives.log.2.gz
/var/log/auth.log
```



Can you complete the following:

- Find all files ending in .py in /usr/lib
- Find all shell scripts denoted by .sh in /var and /usr/lib



Complicated matches

Although not commonly used, the `-o` option allows you to join multiple find statements together. This join is then treated as a logical **OR** operator. This allows for commands such as:

```
$ find /usr/lib -name "*.sh" -o -name ".py"
```

This searches for files in /usr/lib that match either `"*.sh"` or `".py"`.

Filetypes

One of the great things about unix like system is that everything is treated as a file, whether it is a hard disk, a directory, a socket or even a file itself. This is often a major frustration to users familiar with other system. We can use `find` with some addition parameters to be able to restrict our searches to certain types of files. The common parameters that can be passed to the `-type` filter are:

f Search for **files** that are actual files containing information. These are denoted by a **f** in the first place when doing an `ls -la`:

```
-rw-r--r-- 1 root root 1951 Oct 27 06:45 /etc/passwd
```

d Search for **directories**. These contain files and are indicated by a **d** in the first place when doing an `ls -la`:

```
drwxr-xr-x 102 root root 4096 Oct 27 11:45 etc
```

s Search for **sockets**. These are used for inter-process communication and are **not** the same as network sockets. These are denoted by an **s** when doing a directory listing.

```
srwxrwxrwx 1 mysql mysql 0 Oct 25 06:15 /run/mysqld/mysqld.sock
```

l Search for **links**. These are symbolic links which point to a different file. These show up with an **l** in the first position when doing a listing.

```
lrwxrwxrwx 1 root root 28 Oct 20 14:06 /boot/initrd.img -> initrd.img-5.15.0-52-generic
```

Of these the file (**f**) and directory(**d**) options are the most commonly used. Details of others can be found in the [manual page](#). To find all the directories in ones home directory containing *sh* we can restrict our search.

Note: your output may vary depending what files you have in your home directory.

```
$ cd ~
$ find . -type d -name "*sh*"
./ssh
```

Similarly if we wanted to repeat the same but only return files

```
$ cd ~
$ find . -type f -name "*sh*"
./tlscheck.sh
./bashrc
./bash_logout
./bash_history
./lessht
./example.sh
./bin/example.sh
```

Useful examples

You can search for several file types at the same time. Search for files, directories, and symbolic links in the directory /tmp passing these types as a comma-separated list:

```
$ find /tmp -type f,d,l
```

This equivalent to the longer format. The longer format is considered more portable across different unix systems and implementations of `find`.

```
$ find /tmp \( -type f -o -type d -o -type l \)
```

Size

Finding files larger or smaller than a specified size is another common activity for systems administrators, particularly when looking for what is consuming disk space. From a security perspective it can be useful to determine if there are any large files, or which log files are needing attention, or even empty files. the `-size` filter can use less (-) than, more than (+) or exactly **n** units of space. Values are always rounded up. The following suffixes can be used after the numeric value *n*:

b for 512-byte blocks (this is the default if no suffix is used), and is generally relating to the number of sectors used on disk.

c for bytes

w for two-byte words (uncommonly used)

k for kibibytes (KiB, units of 1024 bytes)

M for mebibytes (MiB, units of $1024 * 1024 = 1048576$ bytes)

G for gibibytes (GiB, units of $1024 * 1024 * 1024 = 1073741824$ bytes)

Of these the bytes (**c**) KB (**k**), MB (**M**) and GB (**G**) are the most use practically. Take note that these are based on powers of 2 rather than decimal. Thus a KB is 1 024 (2^{10}) bytes rather than 1 000. We can search for files of exactly 0 bytes locate in /etc using the following. Note the use of `sudo` to ensure we can check all the sub-directories in /etc due to permissions.


```
$ sudo find /etc -size 0c
[sudo] password for cnd:
/etc/newt/palette.original
/etc/apparmor.d/local/nvidia_modprobe
/etc/apparmor.d/local/usr.sbin.rsyslogd
/etc/apparmor.d/local/lsb_release
/etc/apparmor.d/local/usr.lib.snapd.snap-confine.real
/etc/apparmor.d/local/usr.sbin.mysql
/etc/apparmor.d/local/usr.bin.man
/etc/apparmor.d/local/sbin.dhclient
/etc/apparmor.d/local/usr.bin.tcpdump
/etc/.pwd.lock
/etc/security/opasswd
```

We can check some of these using `du` as well as `ls`.

```
$ du /etc/security/opasswd
0      /etc/security/opasswd
$ ls -la /etc/security/opasswd
-rw----- 1 root root 0 Aug  9 11:53 /etc/security/opasswd
```

To find all files on the `/boot` volume greater than 5MB.

```
$ sudo find /boot -size +5M
/boot/System.map-5.15.0-52-generic
/boot/initrd.img-5.15.0-48-generic
/boot/System.map-5.15.0-48-generic
/boot/vmlinuz-5.15.0-48-generic
/boot/initrd.img-5.15.0-52-generic
/boot/vmlinuz-5.15.0-52-generic
$ du -m /boot/initrd.img-5.15.0-48-generic
103    /boot/initrd.img-5.15.0-48-generic
```

We can see that `/boot/initrd.img-5.15.0-48-generic` is actually 103MB in size.

Users and Groups

Being able to determine which files on a filesystem belong to a given user or group is also useful. The `-user` and `-group` filters enable this with `find`. Searching for all files owned by the user `cnd` in `/tmp` is much easier than having to look through all the files. *Note: the files returned may vary depending on your system. these files are left from the earlier experiment.*

```
$ sudo find /tmp -user cnd
/tmp/checks
/tmp/warn
```

We can do some other common searches

```
$ sudo find / -user nobody # searching for files owned by nobody
$ sudo find /usr -user mysql # owned by mysql in /usr
$ sudo find /var -user mysql # owned by mysql in /var
$ sudo find /var -group syslog # files owned by group syslog
```

There are two related special operators which can be used for auditing. These are `-nouser` and `-nogroup`. These commands allow you to find files on your system that are not 'owned' or attributable to any current users or groups (as defined by `/etc/passwd` and `/etc/groups` respectively). This is a common occurrence when users or groups have been deleted, or when archives (especially tar files) have been unpacked incorrectly. To demonstrate this, follow the commands below.

```
$ mkdir /tmp/finddemo
$ touch /tmp/finddemo/nouser
$ touch /tmp/finddemo/nogroup
$ sudo chown :31337 /tmp/finddemo/nogroup # set to unknown group
$ sudo chown 31337 /tmp/finddemo/nouser # set to unknown user
```

```
#sudo used to avoid warning about directories in /tmp owned by root
$ sudo find /tmp -nouser
/tmp/finddemo/nouser
$ sudo find /tmp -nogroup
/tmp/finddemo/nogroup
```

Permissions

Another important component of file metadata is that of the permissions associated with a file. While we typically use these in terms of the textual representation of `-rwxrwxrwx`, `find` works slightly differently with regards to its representation of permissions. Typically their *octal* representation is used. The `ls` command unfortunately does not show these permissions. You can work them out according to the table below.

	2^2	2^1	2^0
R	1		
W		1	
X			1

As a reminder the permissions are additive: So $R+W+X == 4+2+1 == 7$. Thus all permissions for everyone is 777. The numeric mode is from one to four octal digits (0-7), derived by adding up the bits with values 4, 2, and 1. Omitted digits are assumed to be leading zeros. you are encouraged to refer to the manual page for `chmod` for full details.

You can inspect the octal permission of a file using the `stat` command. The example below looks at `/etc/passwd` and `/etc/shadow`. The `stat` command provides some other interesting details that relating to time that we can make use of later.

```
f$ ls -la /etc/passwd
-rw-r--r-- 1 root root 1951 Oct 27 06:45 /etc/passwd
$ stat /etc/passwd
  File: /etc/passwd
  Size: 1951          Blocks: 8          IO Block: 4096   regular file
Device: fd00h/64768d Inode: 525799      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/   root)   Gid: (    0/   root)
Access: 2022-10-27 06:45:30.880295742 +0000
Modify: 2022-10-27 06:45:30.844295988 +0000
Change: 2022-10-27 06:45:30.844295988 +0000
 Birth: 2022-10-27 06:45:30.844295988 +0000
```

```
$ ls -la /etc/shadow
-rw-r----- 1 root shadow 1182 Oct 27 06:45 /etc/shadow
$ stat /etc/shadow
  File: /etc/shadow
  Size: 1182          Blocks: 8          IO Block: 4096   regular file
Device: fd00h/64768d Inode: 525817      Links: 1
Access: (0640/-rw-r-----) Uid: (    0/   root)   Gid: (   42/ shadow)
Access: 2022-10-27 06:45:30.840296017 +0000
Modify: 2022-10-27 06:45:30.832296071 +0000
Change: 2022-10-27 06:45:30.832296071 +0000
 Birth: 2022-10-27 06:45:30.832296071 +0000
```

Note that the octal permissions are actually **four** digits. As explained in the manual:

first digit selects the *set user ID* (4) and *set group ID* (2) and *restricted deletion or sticky* (1) attributes

second digit selects permissions for the *user who owns* the file: read (4), write (2), and execute (1)

third digit selects permissions for *other users* in the file's group, with the same values as the second

fourth digit is for other users not in the file's group, with the same values as the second

To check for certain permissions, the `-perm` filter can be used. This has two modes of operation. The first is the more popular octal based which uses a numeric value such as 0640. This checks to see if a particular bit is set. The secondary mode uses a symbolic mode. Since an exact match is required, if you want to use this form for symbolic modes, you may have to specify a rather complex mode string. For example `'-perm g=w'` will only match files which have mode 0020 (that is, ones for which group write permission is the only permission set). It is more likely that you will want to use the `'/'` or `'.'` forms, for example `'-perm -g=w'`, which matches any file with group write permission. See the [manual page](#) for more details on this. In general it is recommended to use the numeric notation

Applying this we can look for files that have global permissions of `-rwxrwxrwx` or `0777`.

For situations where this kind of detail is overly complex, `find` provides a number of shortcuts that apply to the **current user** that is executing the command.

-executable executable in the context of the current user

-readable executable in the context of the current user

-writable executable in the context of the current user

You can use this to find all *directories* on the system writable by you (as a non privileged user). *Note: the directors will differ on your system*

```
% id
$ id; find / -writable -type d 2>/dev/null
uid=1000(cnd) gid=1000(cnd) groups=1000(cnd),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lxd)

$ find / -writable -type d 2>/dev/null # hide permissions errors with redirect
/var/crash
/var/www/http/uploads
/var/tmp
/proc/265160/task/265160/fd
/proc/265160/fd
/proc/265160/map_files
/sys/fs/cgroup/user.slice/user-1000.slice/user@1000.service
/sys/fs/cgroup/user.slice/user-1000.slice/user@1000.service/app.slice
/sys/fs/cgroup/user.slice/user-1000.slice/user@1000.service/app.slice/dbus.socket
/sys/fs/cgroup/user.slice/user-1000.slice/user@1000.service/init.scope
/run/user/1000
/run/user/1000/gnupg
/run/user/1000/systemd
/run/user/1000/systemd/generator.late
/run/user/1000/systemd/generator.late/xdg-desktop-autostart.target.wants
/run/user/1000/systemd/units
/run/user/1000/systemd/inaccessible
/run/screen
/run/screen/S-cnd
/run/lock
/dev/mqueue
/dev/shm
/home/cnd
/home/cnd/.gnupg
/home/cnd/.gnupg/private-keys-v1.d
/home/cnd/.gnupg/openpgp-revocs.d
/home/cnd/bin
/home/cnd/.ssh
/home/cnd/.cache
/tmp
/tmp/.Test-unix
/tmp/.XIM-unix
/tmp/finddemo
/tmp/.X11-unix
```

```
/tmp/.ICE-unix
/tmp/.font-unix
```



Octal stat!

stat has a mode where it can just return the octal permissions of a file. This can be useful in scripts. Try running the following, which should return 644. Verify that this is correct and what you would expect.

```
stat -c '%a' /etc/passwd
```

Time and Dates

You can also search based on when something was last updated or modified. `find` has a number of filter options relating to time that help one answer questions such as when a file's contents or permissions were last changed.

The more common of these work in time periods of 24 hours. Thus the numeric value *n* passed to the filters below is use as a multiplier to 24. If you want to match the exact 24 hour period you use *n* by itself. More frequently, however, you'll want to say everything since yesterday, or everything "more than 3 days ago." This is accomplished using the `-n` and `+n` options respectively

```
-atime when the file was last accessed
-ctime when the file's permissions were last changed
-mtime when the file's data was last modified
```

For example to look for all files in `/etc` that have been modified within the last 7 days

```
$ sudo find /etc -type f -mtime -7
/etc/shadow
/etc/passwd
/etc/gshadow
/etc/shadow-
/etc/passwd-
/etc/mysql/debian.cnf
/etc/group
/etc/systemd/system/snap-snapd-17336.mount
/etc/ld.so.cache
```

similarly one can look for files that have been modified more than a period of time. Looking for file permissions changed more than 30 days. *Note: your output may vary.*

```
$ cd ~
$ find ~ -type f -ctime +30
/home/cnd/test.cap
/home/cnd/tlscheck.sh
/home/cnd/.profile
/home/cnd/.bashrc
/home/cnd/.bash_logout
/home/cnd/test2.cap
/home/cnd/.gnupg/pubring.kbx~
/home/cnd/.gnupg/pubring.kbx
/home/cnd/.gnupg/trustdb.gpg
/home/cnd/.gnupg/private-keys-v1.d/1E559E8E430FFE87B8299E63BEE8450B8383A176.key
/home/cnd/.gnupg/private-keys-v1.d/82E9CE4AB4FAB84BB199305FAE8A904983CF2CDC.key
/home/cnd/.gnupg/openpgp-revocs.d/EFD62D9A07C4751B42B3BD9C0EB19529D7C1F21D.rev
/home/cnd/.gnupg/mykey.asc
/home/cnd/example.sh
/home/cnd/bin/example.sh
/home/cnd/.ssh/authorized_keys
/home/cnd/test.pcap
```

```
/home/cnd/.cache/motd.legal-displayed  
/home/cnd/.sudo_as_admin_successful  
/home/cnd/testing/script.log  
/home/cnd/testing/hostname.md5
```

There are also minute versions of the `-atime`, `-ctime`, and `-mtime` arguments:

- `-amin` when the file was last accessed
- `-cmin` when the file's permissions were last changed
- `-mmin` when the file's data was last modified

You can also use the `-newer` filter. This compares files against a reference file. This matches if the time of the last data modification of the current file is more recent than that of the last data modification of the reference file. For example `find /etc -type f -newer /etc/shadow` finds any files in `/etc` newer than the file `/etc/shadow`.

3 System Packages

Ubuntu uses a *package manager* to manage all the software components that make up the base operating system as well as any add-on application, tools and utilities that users may install. Keeping track of what file belongs to what package is nearly impossible for a person (the demo system we are using has over 120K files on it!). Fortunately a large part of the package management system is doing exactly this. IF you want to know what package a specific file belongs to you can use `dpkg` or `dpkg-query`.

The following Tasks demonstrate how these tools can be used for the verification and auditing of packages.

Auditing with `debsums`

The `debsums` tool provides an easy way to be able to verify installed Debian package files on the system against checksum lists that are created when the package is installed. These are located in `/var/lib/dpkg/info/`. Currently only MD5 is supported with the details for each package listed in the respective `*.md5sums` files.

You will need to install `debsums` using `sudo apt-get install debsums`.

Once installed we can use the tool to check the integrity of a package:

```
$ debsums wget
/usr/bin/wget                                OK
/usr/share/doc/wget/AUTHORS                  OK
/usr/share/doc/wget/MAILING-LIST             OK
/usr/share/doc/wget/NEWS.gz                  OK
/usr/share/doc/wget/README                   OK
/usr/share/doc/wget/changelog.Debian.gz      OK
/usr/share/doc/wget/copyright                 OK
/usr/share/info/wget.info.gz                 OK
/usr/share/man/man1/wget.1.gz                OK
```

In this case everything verifies as correct.

If we were to modify one of the files:

```
$ sudo mv /usr/share/doc/wget/AUTHORS /usr/share/doc/wget/AUTHORS.orig
$ sudo touch /usr/share/doc/wget/AUTHORS
$ ls -la /usr/share/doc/wget/AUTHORS*
-rw-r--r-- 1 root root 0 Oct 27 10:44 /usr/share/doc/wget/AUTHORS
-rw-r--r-- 1 root root 2331 Oct 27 10:43 /usr/share/doc/wget/AUTHORS.orig
```

Rerunning `debsums` will show an error.

```
$ debsums wget
/usr/bin/wget                                OK
/usr/share/doc/wget/AUTHORS                  FAILED
/usr/share/doc/wget/MAILING-LIST             OK
/usr/share/doc/wget/NEWS.gz                  OK
/usr/share/doc/wget/README                   OK
/usr/share/doc/wget/changelog.Debian.gz      OK
/usr/share/doc/wget/copyright                 OK
/usr/share/info/wget.info.gz                 OK
/usr/share/man/man1/wget.1.gz                OK
```

You can also undertake a broader audit than just each package by running `debsums` without a parameter. This produces a lot of output (>90K lines), which is not very useful on its own. **NOTE** that `debsums` runs in the context of the user executing it. Thus if running as a non privileged user, it will not be able to open and check certain files. Keep an eye on error messages reporting this. For example the kernel package `linux-image-5.15.0-48-generic` and `linux-modules-5.15.0-52-generic`.

...

```
debsums: can't open linux-image-5.15.0-48-generic file /boot/vmlinuz-5.15.0-48-generic (Permission deni
debsums: can't open linux-image-5.15.0-52-generic file /boot/vmlinuz-5.15.0-52-generic (Permission deni
```

```
debsums: can't open linux-modules-5.15.0-48-generic file /boot/System.map-5.15.0-48-generic (Permission
debsums: can't open linux-modules-5.15.0-52-generic file /boot/System.map-5.15.0-52-generic (Permission
...
```

Looking at these files we can see they are restricted to root.

```
$ ls -lad /boot/vmlinuz-5.15.0-48-generic /boot/System.map-5.15.0-52-generic
-rw----- 1 root root 6249017 Oct 13 07:40 /boot/System.map-5.15.0-52-generic
-rw----- 1 root root 11526304 Aug 26 13:13 /boot/vmlinuz-5.15.0-48-generic
```

Executing the command again as a privileged user allows these files to be checked. This is something to be aware of when using the tool.

```
$ sudo debsums linux-image-5.15.0-48-generic
/boot/vmlinuz-5.15.0-48-generic OK
/usr/share/doc/linux-image-5.15.0-48-generic/changelog.Debian.gz OK
/usr/share/doc/linux-image-5.15.0-48-generic/copyright OK
```

Returning to the general operation of debsums, we can split the output into two files: /tmp/checks for the output and /tmp/warn for the warnings. you will see there is no output to screen. This is because the *STDERR* output stream (denoted by stream 2 in the bash shell) has been redirected to /tmp/warn. This command can take a few minutes to run.

```
$ sudo debsums > /tmp/checks 2>/tmp/warn
```

The checks file can be verified to see what is reported

```
$ cat /tmp/checks | grep -v "OK$"
/usr/share/doc/wget/AUTHORS FAILED
```

This as expected returns that only one file does not match its recorded checksum. This is the file we modified earlier so is not unexpected. you can do further filtering on this file to restrict by any other criteria you may have.

It is not always required to go through this whole process, as debsums has a variety of additional options. You can explore these by using the manual page and the `--help` parameter.

```
$ debsums -csx 2>/tmp/warns
/usr/share/doc/wget/AUTHORS
```

Exploring missing files is outside the scope of the usage we need for the CND course.

We can look at the type of file format that the hash files used by debsums are, in this case for the wget package:

```
$ cat /var/lib/dpkg/info/wget.md5sums
d4dbfd931aeb9931ba381178ad636e4f usr/bin/wget
3ed8e143eb4e79899c150f201d86bdb7 usr/share/doc/wget/AUTHORS
e6f406888c940a31427b6eaccabf4b62 usr/share/doc/wget/MAILING-LIST
54de5fcb21cda2033ee2ce5d64227522 usr/share/doc/wget/NEWS.gz
c18e7472129a0cb1b0c5c0a08a39a69e usr/share/doc/wget/README
38fa6e7028478f6e7bc4677739f371da usr/share/doc/wget/changelog.Debian.gz
2e3239ba8aedc57b0114889b4bbf2f7d usr/share/doc/wget/copyright
cce4ca2ff9a79fffc5944a9518a8f6a4 usr/share/info/wget.info.gz
ded89709bf7d87bf33cb8ce1c20175ce usr/share/man/man1/wget.1.gz
```



The hashes above may not match exactly with what is presented here, as software may have updated since this was produced. The principles remain the same.

We can see these are very similar to those produced by tools such as `md5sum`. We can use such files directly with `md5sum`. Not that there is no leading / in the paths. This means that the paths are treated as **relative** rather than **absolute**. This means that it is important be in the file system root if making use of these files.

```
$ cd /
$ md5sum -c /var/lib/dpkg/info/wget.md5sums
usr/bin/wget: OK
usr/share/doc/wget/AUTHORS: FAILED
usr/share/doc/wget/MAILING-LIST: OK
usr/share/doc/wget/NEWS.gz: OK
usr/share/doc/wget/README: OK
usr/share/doc/wget/changelog.Debian.gz: OK
usr/share/doc/wget/copyright: OK
usr/share/info/wget.info.gz: OK
usr/share/man/man1/wget.1.gz: OK
md5sum: WARNING: 1 computed checksum did NOT match
```

You should restore the file back to its original state using the commands below. Then verify that the `wget` package passes.

```
$ sudo mv /usr/share/doc/wget/AUTHORS.orig /usr/share/doc/wget/AUTHORS
$ debsums wget
```



Is this secure? Consider what you have discovered exploring this system and using the `debsums` command above. Given this context, discuss the following points with other and make some notes as to your consensus.

- Does a system like this actually provide any meaningful security?
- How does this compare to tools previously considered such as `rkhunter`?
- What are some of the risks of such tool in the event of a potential system compromise?
- What else should one consider when storing hashes local to the system?
- What recommendations could you make to improve the overall security?



MD5 deprecated

You may wonder why Debian (and related derivative distributions such as Ubuntu and Mint) still used MD5 for checksums. This is a fair point and discussion has been ongoing since 2009 around this, but does not seem to have been [resolved](#) as of October 2023. Many other packaging systems have moved to using SHA256. While MD5 has a number of known issues, for the purposes of this course we can treat it as valid, and thus can use it for validating files.

Auditing with dpkg

The core of the package management system used by Debian family operating systems is `dpkg`. This tool provides a number of way to get information about the state of the system and the packages installed on it. In section 2 you saw the demonstration of how to determine which package a file belonged to. This Task demonstrates a number of other useful features that the tool offers.

Listing installed packages

When checking servers, it is useful to determine what packages are installed. There are various reasons for this, ranging from wanting to determine what is needed to build a new server, to verifying that packages such as compilers are not installed as part of a hardening process. This is also a good way of determining the version of packages, particularly when they have been manually installed rather than from the public package repositories. This can be done as follows:

```
$ dpkg -l
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                                                    Version                                           Architecture >
+++-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+----->
ii adduser                                                  3.118ubuntu5                                     all                                               >
ii amd64-microcode                                          3.20191218.1ubuntu2                             amd64                                             >
ii apache2-bin                                              2.4.52-1ubuntu4.1                               amd64                                             >
ii apparmor                                                 3.0.4-2ubuntu2.1                                amd64                                             >
ii apparmor-utils                                           3.0.4-2ubuntu2.1                                all                                               >
ii apport                                                   2.20.11-0ubuntu82.1                             all                                               >
ii apport-symptoms                                         0.24                                              all                                               >
ii apt                                                       2.4.8                                             amd64                                             >
ii apt-utils                                                2.4.8                                             amd64                                             >
....
```



By default the output of `dpkg -l` is put through a *pager*. This allows you to scroll up and down, as well as left and right using the arrow keys on the keyboard. The ">" on the left hand side indicates there is more text available.

To validate if a specific package is installed you can specify the name. The response here is slightly different, and included a brief description.

```
$ dpkg -l wget
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name           Version           Architecture Description
+++=-----+-----+-----+-----+
ii  wget             1.21.2-2ubuntu1  amd64         retrieves files from the web
```

Checking installed packages

Checking for the presence of a package is also important. similarly to how we were able to search for files as shown previously a variation of that command can be used `dpkg -s package`. Take note of the *lower case*.

```
$ dpkg -s wireshark
dpkg-query: package 'wireshark' is not installed and no information is available
Use dpkg --info (= dpkg-deb --info) to examine archive files.
```

```
$ dpkg -s wget
Package: wget
```

```
Status: install ok installed
Priority: standard
Section: web
....
```

The first command shows the response when a package is not found, and the second when it is.



Error checking Many unix utilities have a *return code* which they return to a shell after executing. By convention a value of 0 means all okay, with values of 1 or 2 typically indicating various error states. In the `bash` shell this response is stored in the environment variable `$?` . This can be used in scripts to change program flow based on the success/failure of a previously executed command. You can check this variable by typing `echo ?` . Try repeating the two `dpkg` commands above and checking the *return code* after each.

Locating files

The inverse to what we saw previously is going from a package name to a listing of the files that it installed.

```
$ dpkg -L wget
/.
/etc
/etc/wgetrc
/usr
/usr/bin
/usr/bin/wget
/usr/share
/usr/share/doc
/usr/share/doc/wget
/usr/share/doc/wget/AUTHORS
/usr/share/doc/wget/MAILING-LIST
/usr/share/doc/wget/NEWS.gz
/usr/share/doc/wget/README
/usr/share/doc/wget/changelog.Debian.gz
/usr/share/doc/wget/copyright
/usr/share/info
/usr/share/info/wget.info.gz
/usr/share/man
/usr/share/man/man1
/usr/share/man/man1/wget.1.gz
```

The output here may seem a little strange as it lists **both** directories and files. Thus you can see the root directory (`.`) as well as `/etc` , `/usr` and others. The reason for this is that the package needs to create these if they do not already exist.

Auditing packages

Despite what may be implied by the naming, the `--audit` parameter to `dpkg` does not perform a file integrity audit. Rather this command audits that package dependencies are met, along side version compatibility and any listed conflicts. Running this is a good part of standard operation, but offers little in the way of value in terms of security, and detecting possible compromise of a system.



Cruft

`cruft` is another auditing tool that describes itself as:

a tool to look over your system for anything that shouldn't be there, but is; or for anything that should be there, but isn't.

This can provide a deeper dive into checking a running system against what is expected based on the data contained in the package manager repository in `/var/lib/dpkg/info/`.

4 Deliverable

The deliverable items this week are skills development. You are expected to have worked through the tasks and ensure that you understand and can undertake the following:

1. Understand the basic format of a syslog style file
2. Be able to access and read system logs
3. Understand how Logs can be processed using common text processing tools - this ties back to the tasks in the first tutorials and regular expressions
4. Validate the integrity of packages on an Ubuntu system
5. List packages installed on an Ubuntu System
6. Find files on a system.
 - Given a filename or a pattern you should be competent to return the full path in which this is found
 - Given a filename or path, determine if this was installed by a package, and if so what one?
7. Demonstrate and use the various modes of the `find` utility



Required Reading

You should read the following articles, which are in scope for the examination and the Engagement activity 3:

- Forte, D. V. (2004). The “ART” of log correlation: part 1: Tools and techniques for correlating events and log files. *Computer Fraud & Security*, 2004(6), 7-11. [10.1016/S1361-3723\(04\)00075-2](https://doi.org/10.1016/S1361-3723(04)00075-2)
A similar version that you may find easier to read: [Alternate Version](#)
- Sabato, S., Yom-Tov, E., Tsherniak, A., & Rosset, S. (2007, January). Analyzing system logs: A new view of what’s important. In 2nd USENIX workshop on Tackling Computer Systems Problems with Machine Learning Techniques. [Link](#)
- Nawyn, K. E. (2003). A security analysis of system event logging with syslog. SANS Institute, no. As part of the Information Security Reading Room. [Link](#).
This is a fairly high level review of the syslog format and some challenges.

There is a deliberately light load this week to allow people to spend some time on the exam preparation, and to enable catch-up for those that are still struggling with networking.

A News

The following are new articles of interest in the last week. These are all relatively short, and you should get in the habit of staying aware of what is going on in the industry at large. The topics discussed here fall in scope for the engagement tasks.



News Week of 9 October 2023

DDoS [How it works: The novel HTTP/2 'Rapid Reset' DDoS attack](#)

Google services and Cloud customers have been targeted with a novel HTTP/2-based DDoS attack which peaked in August. These attacks were significantly larger than previous attacks. Google Response Team helped lead a coordinated disclosure process with industry partners to address the new HTTP/2 vector across the ecosystem.

ATTACK [Everest cybercriminals offer corporate insiders cold, hard cash for remote access](#)

The Everest ransomware group is stepping up its efforts to purchase access to corporate networks directly from employees amid what researchers believe to be a major transition for the cybercriminals. Everest said it will offer a portion of the profits generated from successful attacks to those who assist in its initial intrusion.

BREACH [Air Europa Asks Customers to Cancel Cards After Breach](#)

A leading airline has told some of its customers to cancel their payment cards after revealing their details were compromised in a data breach.

MALWARE [ShellBot DDoS Malware Installed Through Hexadecimal Notation Addresses](#)

The ShellBot malware is being installed on poorly managed Linux SSH servers, with recent cases confirming its use of hexadecimal IP addresses to evade behavior-based detection.

APT [ToddyCat: Keep calm and check logs](#)

ToddyCat is an advanced APT actor. The group has been responsible for multiple sets of attacks against high-profile entities in Europe and Asia. In this article, we'll describe their new toolset, the malware used to steal and exfiltrate data, and the techniques used by this group to move laterally and conduct espionage operations.

B Online Skill Builders

[Cyber Security Challenge UK.com/](#) provides a number of interesting online skill builders and challenges. Two that may be of interest for those of you still needing some re-enforcement around your networking knowledge are listed below.

These activities are **not** part of the formal curriculum, but are provided as a means to help those who are still struggling with the concepts.

Network Ports

Task: [Network Ports](#)

Explore and test your skill relating to the use, identification and role of ports when looking at networking

Task: [Firewall Introduction](#)

This provides an introduction to a simple firewall and how it operates.