Cybersecurity

Mathias Ritter

Oli's Part

Introduction

- Welcome to Cybersecurity
- Why do we need this? Let's just take a look at the news…

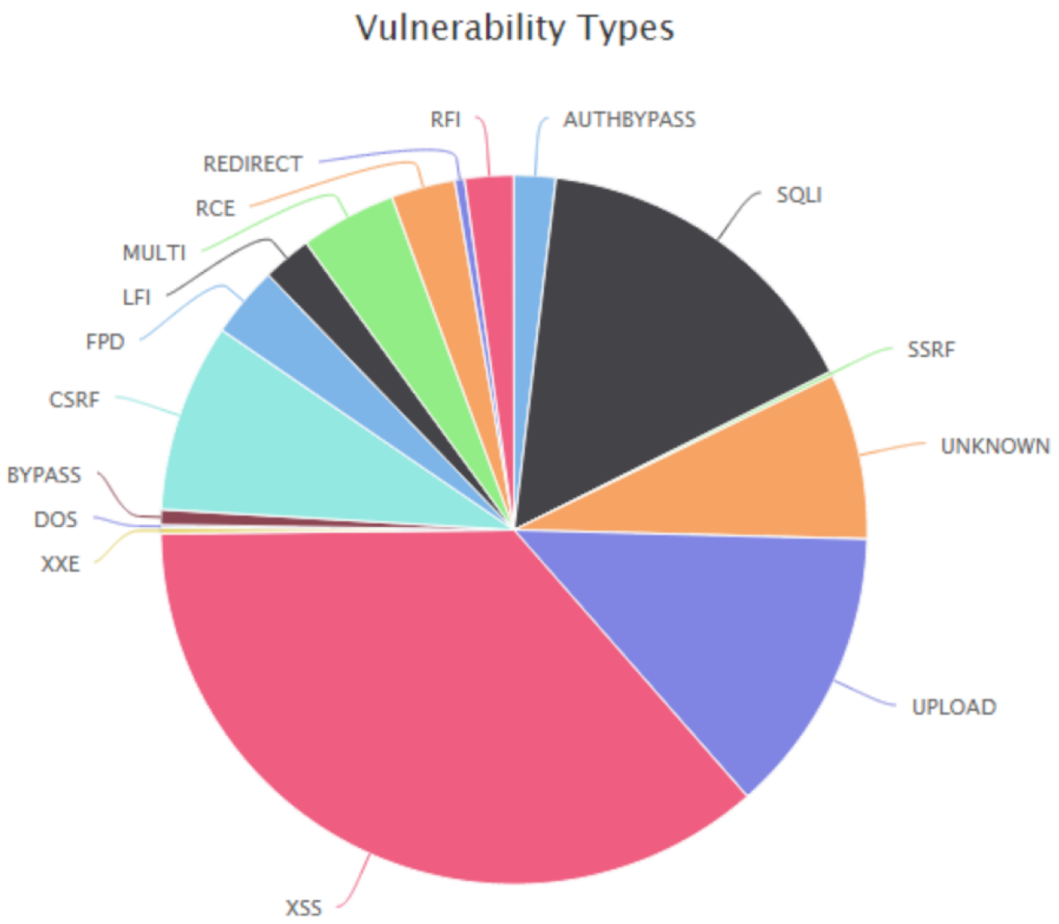
- US voting machines are easy to hack
    - Vulnerabilities were reported over a decade ago but are still present
    - Machines have accessible USB ports and are running Windows XP, so you can plug in a USB drive with an auto executable file and it will run

- Unfixable iPhone exploit re-opens world to jailbreaking
    - You can get root access to the system and install any software you want
- Are security exploits a good or a bad thing?
    - It's good if you are a security researcher and you need to gain more (i.e. root) access to the system
    - It's bad if you download some software and it includes malware that can get the same access

- Iraq's government websites were hacked
    - In total almost 30 websites were attacked

- Tory Party app leaks private attendee information
    - It allowed you to log in as other people with simply using their email addresses
    - E.g. you could log in as Boris Johnson and gain access to all personal information and even make modifications

- Now, let's talk about the most popular CMS WordPress (WP)
    - CMS stands for Content Management System
    - That's basically something a lot of people use to make their own blog
- It's a system full of vulnerabilities!
    - About 55% of all vulnerabilities are from plugins, 30% are from the WP core and 15% are from themes
- What are the most popular vulnerability types? Let's look at the following graphic



- If you don't believe me, take the WordPress challenge
    - Get a new domain
    - Install WP as well as the top 10 featured plugins and the top 3 themes
    - Wait a few days and your website will be hacked!
- So the TLDR is: just say no to WordPress!

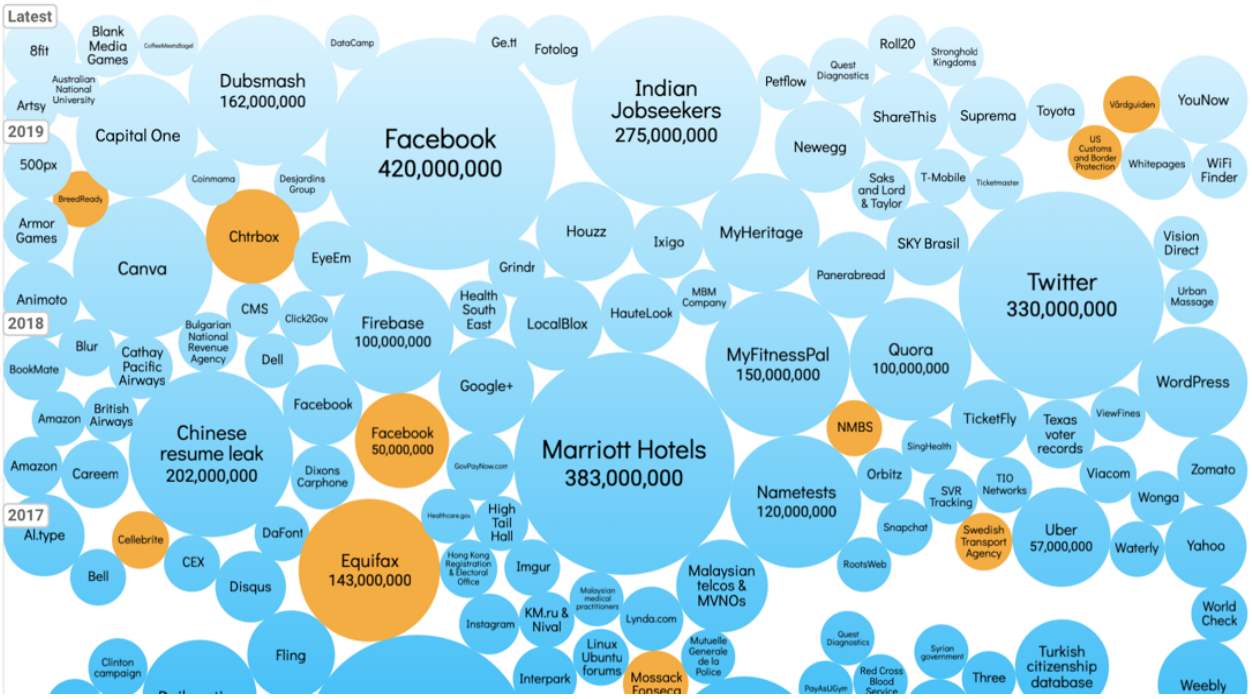
Just Say No!

- News shows importance of cybersecurity
    - Read it and stay up to date
    - Cybersecurity doesn't exist in a bubble
    - As an employee you can use past incidents to argue that the company needs to invest a lot of money to protect themselves against attacks

- What is cybersecurity all about?
    - It's about electronic data and abuse and misuse
- Security is still an IT system, so you have to consider
    - Technology
    - Procedures and
    - People
- All of those 3 need to be addressed to make a system secure

- Cyber security is about electronic data abuse and misuse
- Security is still an IT system
    - It's about technology, procedures and people
    - An example for a bad procedure is having one shared drive with read & write access for all employees in a company

- As we have just seen the news with all those hacks, the question is: Which company can you still trust?
    - Actually not many, as you can see in this infographic: https://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/



    - These are just the more recent attacks, if you go to the page and compare them to older attacks you can see that attacks get a lot more frequent and bigger

- You might think that an attack is not bad if they just got a small amount of information from you, like name, address, email, telephone number and location.
    - However, you can do a lot of damage with this information
    - For example, suppose an attacker gets access to your email account
    - How many other accounts can they reset with sending a link to your email?
    - Exactly, probably almost every account
    - Even if they don't have access to your email account, they could probably just ring up companies and ask for the account to be reset
    - They have your name, address, and maybe your date of birth and that's enough for most companies to believe that the attacker is you!
    - So the TLDR is: stealing your identity with this information is easy

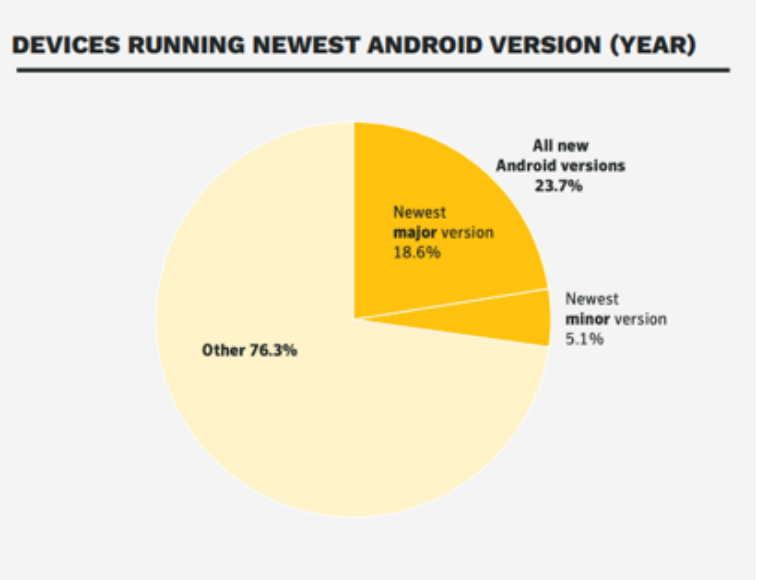
- If the news wasn't enough for you yet: there is a lot of more sadness out there
    - In one of the university system, this is how the login is performed:

```php
//Perform login
$u = $_POST['username'];
$p = $_POST['password'];
$l = mysqli_query("SELECT * FROM users WHERE username='$u' AND password='$p'");
if ($user == mysql_fetch_array($l)) {
    //Succesful login
```

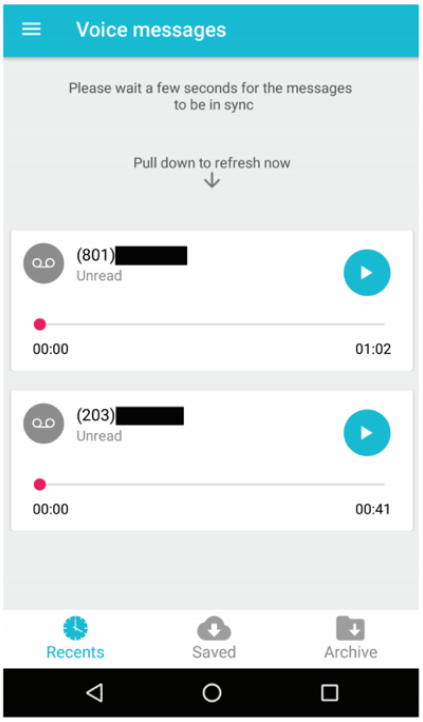
    - One could easily perform an SQL injection to log in as an arbitrary user
    - I'll explain SQL injection later, but basically you can manipulate the query by providing a certain username/password
    - Disclaimer: Do not try to find out which university system this is, or you can get into trouble…

- What would happen if your phone was hacked?
    - All your personal info can be stolen
    - Access to all your accounts, because you can reset passwords via email (like I mentioned above) and your emails are on your phone
    - You could be blackmailed
    - The attacker could pretend to be you
    - If you have access to some business (because you are an employee) this data is at risk as well

- Many people don't update their mobile phone OS, making it easier for hackers to exploit vulnerabilities
    - Android phones tend to be more out of date than iPhones
    - Pegasus spyware gives you remote access to any phone without the user noticing anything on their phone
    - There are probably even a few people who read these notes right now and have malware running on their phones without knowing it...



- Let's look at some more numbers that are related to cybersecurity
- 3.2 billion
    - This is the estimated amount of current internet users
    - This number keeps growing and growing
- 30 million
    - This is the number of mobile malware apps in total
    - There is still a steady growth of new malware
    - They are getting clever: Attackers sending a custom tailored email to you so you think it's really from your provider
    - They claim you should download a voice mail app to listen to a message
    - Once you install the app it has full access to your network
- 19%
    - This is the amount of websites powered by Wordpress, sadly
- 320 hours
    - This is the duration of the longest DDOS attack ever
- Wordpress
    - … is never the answer to anything!



- How is cybersecurity defined?
    - It is the detection, protection and prevention against digital threats and attacks
    - As mentioned above, we have to consider technologies, procedures and people - basically the entire digital ecosystem

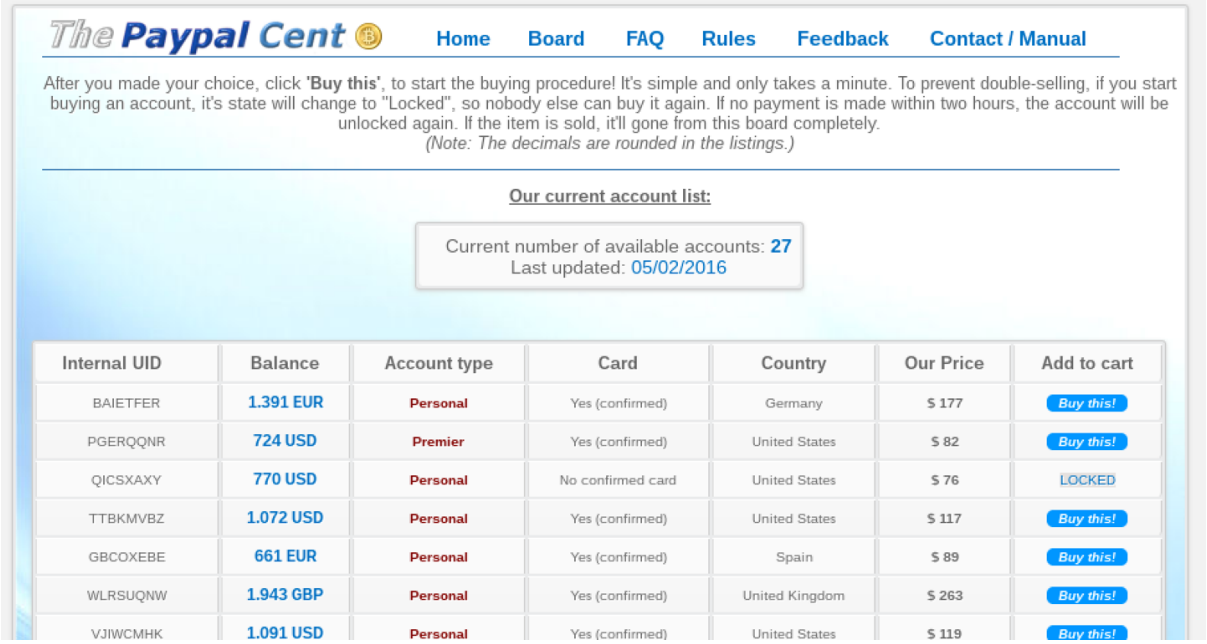
- What kind of threats are there?
    - Visual surveillance - someone watches you when you enter your password
    - Wiretapping - a software logs your keyboard inputs and sends them to the attacker
    - Key interception
    - Impersonation - you log into a website that looks like the real one but it's fake and from the attacker
    - Social engineering
    - Physical security
    - Malware
    - Combinations of attacks

- Who is affected by those threats?
    - Basically everyone can be...
    - National security and infrastructure
    - Government and public bodies
    - Banks
    - The everyday citizen including you!

- Who are the attackers?
    - Criminals or terrorists
    - Hackers or ethical hackers
    - Employees, former employees, contractors, competitors or customers
    - Government or whistleblowers

- Remember that a system is only as strong as its weakest link
    - There is no point in building a massive fence around your house if you leave a gap
    - You are the weakest link, goodbye!

- What does the future hold?
- In the future, we'll have even more systems that can be attacked and data that will need to be protected
- Just think of the Internet of Things
    - If you don't know what that is: it basically means connecting all different sorts of systems to the internet
    - Examples are home appliances, vehicles, medicine, assistants
- The number of users is growing constantly
- Digital technologies are replacing physical things
    - You don't need to have the physical debit card and the PIN when you can just make a payment with the card number online…
    - Attackers are selling card details and paypal accounts online



The Paypal Cent

Home Board FAQ Rules Feedback Contact / Manual

After you made your choice, click '**Buy this**', to start the buying procedure! It's simple and only takes a minute. To prevent double-selling, if you start buying an account, it's state will change to "Locked", so nobody else can buy it again. If no payment is made within two hours, the account will be unlocked again. If the item is sold, it'll gone from this board completely.
(Note: The decimals are rounded in the listings.)

Our current account list:

Current number of available accounts: **27**
Last updated: 05/02/2016

Internal UID	Balance	Account type	Card	Country	Our Price	Add to cart
BAIETFER	1.391 EUR	Personal	Yes (confirmed)	Germany	$ 177	Buy this!
PGERQQNR	724 USD	Premier	Yes (confirmed)	United States	$ 82	Buy this!
QICSXAXY	770 USD	Personal	No confirmed card	United States	$ 76	LOCKED
TTBKMVBZ	1.072 USD	Personal	Yes (confirmed)	United States	$ 117	Buy this!
GBCOXEBE	661 EUR	Personal	Yes (confirmed)	Spain	$ 89	Buy this!
WLRSUQNW	1.943 GBP	Personal	Yes (confirmed)	United Kingdom	$ 263	Buy this!
VJIWCMHK	1.091 USD	Personal	Yes (confirmed)	United States	$ 119	Buy this!

- There are going to be more infrastructure attacks
    - Malware can be used to attack oil, gas, electric grid
    - Healthcare, finance and energy systems can be attacked as well
    - University attacks are growing
- States are getting more and more involved
    - E.g. USA, Russia, China, …

- What are new solutions to threats?
    - We can focus on resilience - accept that there sometimes isn't a solution and reduce the impact when a breach occurs
    - We can outsource the problem - move it to the cloud and it's someone else's problem
    - We can send phishing emails to our own employees to test if they can spot them - otherwise we need additional training for our employees
    - We can use artificial intelligence to discover weaknesses
    - We can get more people involved

- For now, here is what you need to do
    - You need to find your (the system's) weaknesses
    - Yes it is going to take time and money
    - But it's better to find it yourself (penetration testing) rather than having someone else find it and attack your system

- How do we deal with threats?
    - We use security policies and mechanisms to keep a system secure
    - We also use risk management
    - What is the probability that an attack occurs, and what would be its impact?
    - Once we identified this for all threats we need to figure out how we want to control the risk

    - We can try to avoid it, we can minimise it and/or we can define what to do to mitigate it in case the attack occurs
    - Here is an example

System	Customer database linked to online shop
Risk	Customer details could be stolen by attacker
How to avoid?	Encrypt database so only encrypted details could be stolen
How to minimise?	Penetration testing to reduce risk of public vulnerabilities
Mitigation if it occurs?	On detecting a breach, immediately shut down site

- What are security policies?
    - A security policy is a statement of what is or is not allowed
    - We want to ensure confidentiality, integrity, availability and authenticity
- How do we ensure security policies?
    - We use principles and guidelines
    - More on this topic later

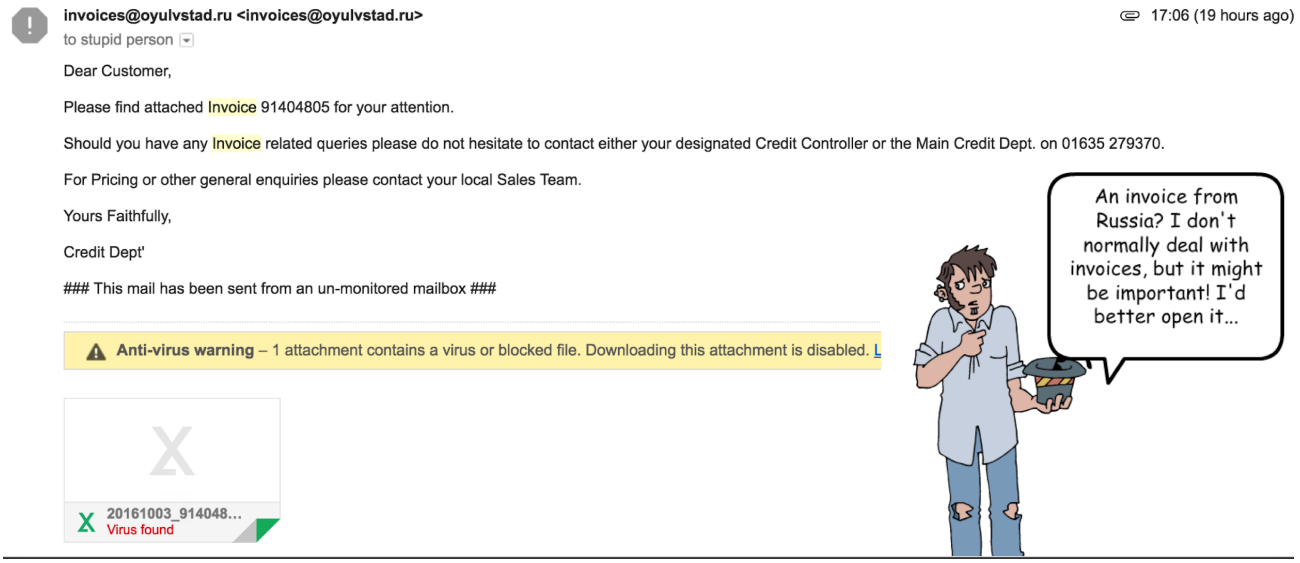
- What kind of security mechanisms can we use?
    - Authentication - something like a login
    - Authorisation - is the user allowed to perform this action?
    - Encryption
    - Firewalls
    - Network segmentation - don't put everything in the same network
    - Backups
    - Updates and patches
    - Training and education

- How can we detect a threat?
    - Hashes and digital signatures - more on this topic later
    - Intrusion detection systems - detect that an attacker has made it into a network
    - Malware/antivirus scanning
    - Monitoring systems
    - Honeypots - some fake system that acts as a trap
    - Disclosure policies

- Why are backups good?
- There are many reasons but since this is cybersecurity let's look at ransomware
    - Ransomware encrypts all your files and demands a payment from you to decrypt them
    - If you have a backup you can just restore the files
    - If you don't have a backup you are basically screwed
- An example of ransomware is Cryptolocker

- It is estimated that it raised over 30 million dollars in 2013
- 43% of all users in a survey said that they had paid the fine to decrypt their files

- Why is it a good idea to train the users?
- Well, take this scenario for example



- How do you get cybersecurity right?
  - Don’t make assumptions

- Don’t be a perfectionist
- Remember the users of the system
- Engage with the users
- Build up multiple layers of security so that if one fails you still have other layers
- KISS - no, not the guy next to you - it stands for Keep It Simple, Stupid
- Be realistic
- Don’t keep secrets - security through obscurity is evil
- Remember that the weakest link defines how secure your system is - so try to keep it as strong as possible

Web Applications

- First, a little news update
  - FIFA: personal information of 1,600 people was leaked through a signup form because it was pre-filled with other people’s data in some cases
  - Toms shoes website was hacked and the attacker had access to the whole company network including info about their customers

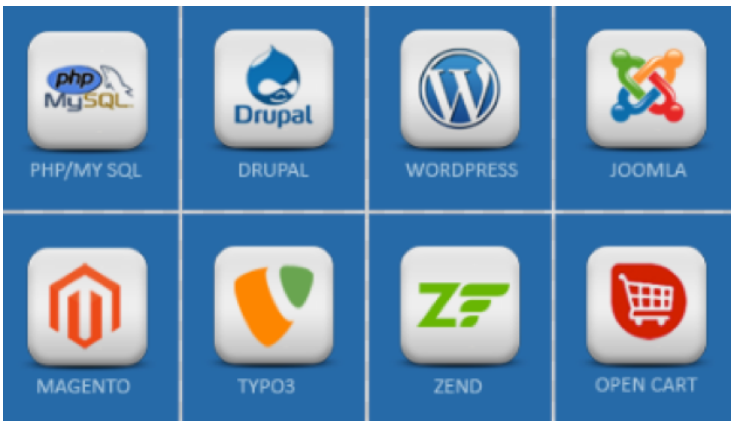
    - 3 American hospitals infected by malware so they had to turn away non-emergency patients
- All of these things could have been easily prevented by using up-to-date software, proper procedures and penetration testing

- What does the average user do on your web application?
    - Follows the steps through
    - Enters the right information and clicks the right buttons
    - Always puts in the correct information at each stage
    - Follows the process through from the beginning
    - Basically, they just do everything in the way you would expect as a developer
    - Now here is the bad news: they don't exist!
- In comparison, here is the non-average user
    - Doesn't follow any of the steps
    - Presses the worst buttons at the worst possible time
    - Inputs the incorrect information at each stage
    - Ignores any process you've tried to put into place
    - Basically, they do everything wrong that can be done wrong
    - Unfortunately, these users do exist, and sometimes you need to be one of those to find security vulnerabilities

- 10 years ago websites were full with vulnerabilities
    - They were constantly being hacked, personal data was stolen
    - Cross-site scripting, SQL injection, no user input validation etc, it was all in there!
- So you think it's all water under the bridge?
- Unfortunately, websites today still have the exact same problems
    - Yes you read that correctly, it means that a lot of developers haven't learned anything from the past 10 years
    - We still face the same vulnerabilities nowadays

- Alright, so what exactly do you mean by "web application"?
    - It's not just websites you can view and interact with in your browser
    - We count everything that communicates via the web (e.g. using HTTP)
    - More examples are APIs, mobile apps and other internet of things devices

- There is a lot of off the shelf web software that anyone can use to host a web application, check out the graphic below for examples



- Have you spotted it? Of course, WordPress is one of those as well 😨
    - If you don't know why it's bad, read the introduction bit of the notes

- So we mentioned communication using HTTP. How does that work?
- HTTP is used to send requests to the server, and to receive back some information in responses
    - HTTP requests and responses contain of a header and a body section, however, the body can be omitted
    - The body usually contains the HTML from the server or any user inputs from the client (but it can be empty as well as mentioned)
    - The body is not the only place in which information can be transmitted - more on that later
    - This is how an HTTP request looks like - this one has just a header and no body

```
GET / HTTP/1.1
Host: google.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

    - And here is the response - you can see the header and below the beginning of the body which is separated from the header by a blank line

```
HTTP/1.1 200 OK
Date: Tue, 14 Aug 2018 08:21:32 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Strict-Transport-Security: max-age=86400
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2018-08-14-08; expires=Thu, 13-Sep-2018 08:21:32 GMT; path=/; domain=.google.com
Alt-Svc: quic=":443"; ma=2592000; v="44,43,39,35"
Connection: close
Content-Length: 221306

<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en-GB"><head><meta conte
href="/images/branding/product/ico/googleg_lodp.ico" rel="shortcut icon"><meta content="origin" name=
nonce="vigWecs4/gQKZltTgjGUzQ==">(function(){window.google={kEI:'DJFyW8OiGuPjkgW39JvQCg',kEXPI:'31',
ction(){return(new Date).getTime()};google.timers={};google.startTick=function(c,b){var
```

- Let's look at the first line of that HTTP request above since it's the most important one

GET	GET denotes the type of the request (more on that below)
/	/ is the URI (Uniform Resource Identifier). You basically have different URIs for different information so you can tell the server what you want. They work just like paths, so the slash denotes the root of the web server.
HTTP/1.1	HTTP/1.1 denotes the protocol and the version

- Now, let's look at the first line of the response

HTTP/1.1	HTTP/1.1 denotes the protocol and the version
200 OK	200 is the status code and OK is the description of it. So the status code 200 basically means that the server says everything is OK, here is your response.

- A quick guide to status codes
  - 200 OK: request was successfully served
  - 4xx: client side error - It is your fault, you sent a wrong request to the server
  - 5xx: server side error - It is not your fault, something went wrong on the server

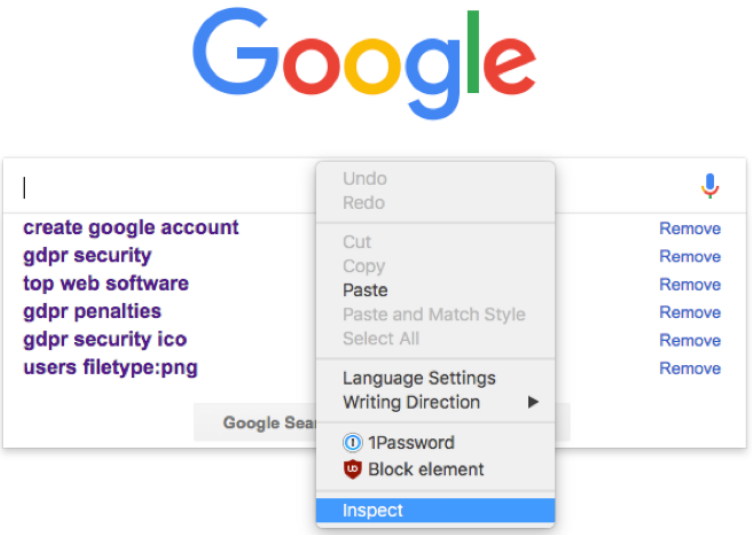
- There are different types of HTTP requests, here are the most important ones

Request Type	**What it should be used for**	**Example**
GET	Query a resource	Get a user's profile data
POST	Create a resource *Sometimes also: Update a resource*	Create a new user *Sometimes also: Update a user's profile*
PUT	Update a resource	Update a user's profile
DELETE	Delete a resource	Delete a user

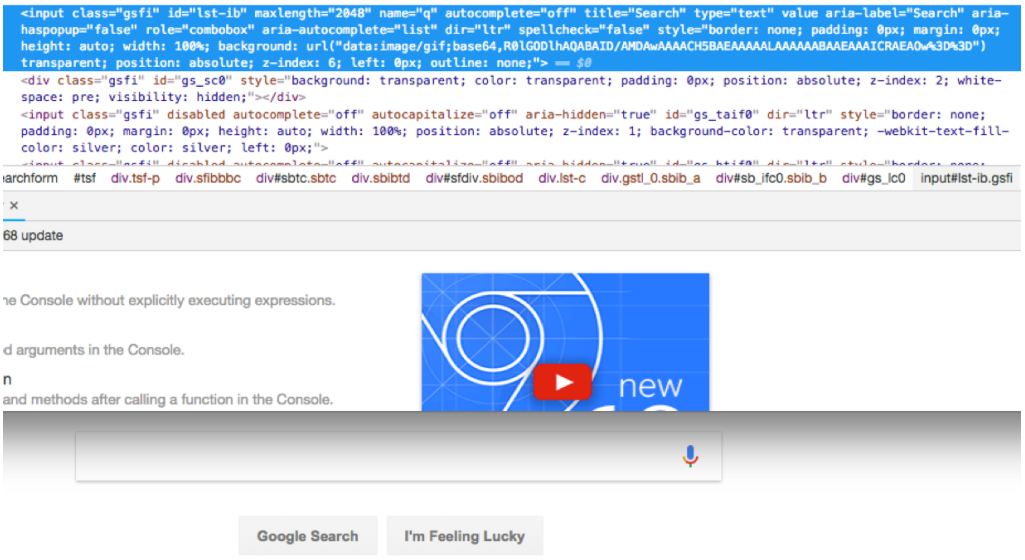
- One thing to note is that these are only conventions
  - You as a developer should follow them
  - However, you could still implement something like deleting a user using a GET request
- When you enter a URL in your browser and hit enter, your browser will send a GET request

- When you click a hyperlink to navigate to another page, it will be a GET request as well
- When you submit a form, it is usually a GET or a POST request
- With JavaScript you can send any sort of request, e.g. on clicking a button

- The body of an HTTP response usually contains some HTML
    - If you want to view the HTML code of a website, right click and select inspect



    - And you will see that a window like this will open



- In a response the data can only be in the header and body, however, you can transmit information in a request in more places
- In total there are 4 places where you can transmit information in a request, as shown with examples in the table below

Where	Example	Data	Structure	Purpose
URI	/users/5	5	Values in the	View the user

parameter			URL separated by “/”	with the id “5”
Query parameter	/search?q=Southampton	q=Southampton	Key value pairs separated by “&”	Search for “Southampton”
Body	{ name: “Oli” }	E.g. JavaScript object (JSON)	Can be anything as long as the server knows how to understand it	Create a new user with name “Oli”
Header	Authorization: Basic QWxhZGR	E.g. a string	Key value pairs separated by “:”	Send a token for authorisation

- Query parameters are also called GET parameters
- Parameters in the body are also called POST parameters

- So without further ado, let’s look at the most common vulnerabilities

Vulnerabilities

Information Exposure

- It basically means accidentally leaking some information

- For example, you left a test file on the server
  - testing.php
  - phpinfo.php
  - info.php
  - debug.php
- Or, you left backup/swap files on the server such as
  - index.bak
  - ~index.php
  - .index.php.swp
  - index.zip

- How can you discover it? Here are some things to look out for
  - Is there anything in the headers?
  - Can you find anything in the HTML source code?
  - Can you find any other directories?
  - Can you find any testing files?
  - Can you find any backup files?

- How do you fix it?

    - Well just don't leave those files on the server
    - Check all directories for any files that shouldn't be there

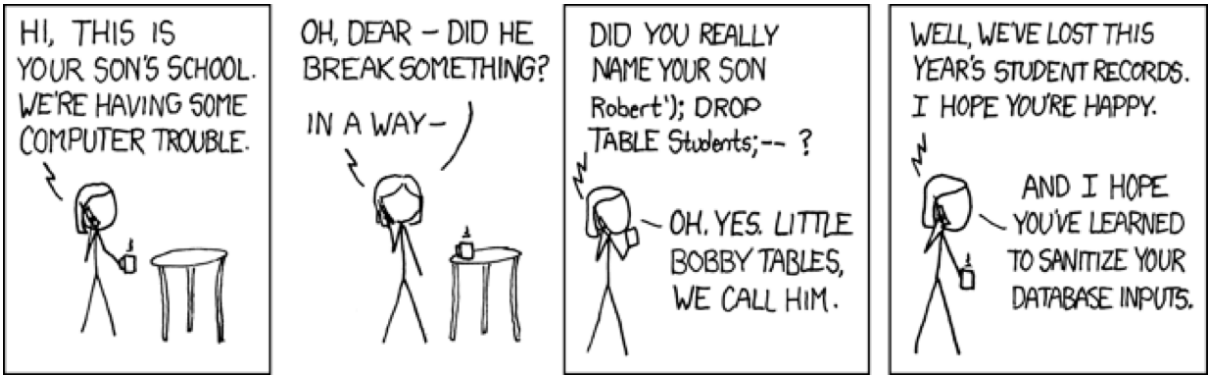
SQL Injection

- Before we talk about SQL injection a brief introduction to how a web application uses SQL
- SQL is used when an application uses a database
    - It's a way to retrieve data from the database
    - You can also add or update data
    - So most of the time we use SELECT, INSERT, UPDATE statements to achieve this

- A SELECT request is used to query the data and usually contains the following clauses
    - SELECT columns
    - [INTO new table]
    - FROM table or view
    - [WHERE specific rows or a join is created]
    - [GROUP BY grouping conditions (columns) – HAVING group-property (specific rows)]
    - [ORDER BY ordering criterion ASC | DESC]
- Here are some examples
- SELECT * FROM Table
    - * Means all columns
- SELECT Column,Column2… FROM Table
    - Specify specific columns
- SELECT Columns FROM Table WHERE Conditions
    - Specify conditions on results returned
- SELECT Columns FROM Table LIMIT Number
    - Limit the number of results
- SELECT * FROM Table ORDER BY Column ASC/DESC
    - Specify an ordering

- The part where it gets interesting is when user information directly goes into a SQL query
- For instance, say you are developing a page that allows to view some user's profile by username
    - There is going to be some input provided by the user of your application that determines which profile is going to be displayed
    - This information will go into the SQL query
    - Suppose the query is SELECT * FROM users WHERE username="*<USERNAME INPUT>*"
- Now, if the input provided is Bob

    - The query will become SELECT * FROM users WHERE username="Bob"
    - This is fine and will work
- However, suppose someone provides the input Bob"; DROP TABLE users; --
    - Then the query will become SELECT * FROM users WHERE username="Bob"; DROP TABLE users; -- "
    - This will delete the entire user table!
    - Why? The SELECT query gets executed but we appended another query that deletes the table
    - At the very end we use the double dash to comment out the remaining quotation mark



- Here is another example - take the login functionality of a web app
- The user provides a username and password and the application will look for a match in the database
- Here are 3 inputs, the first one is okay and the other two use SQL injection to bypass the password check

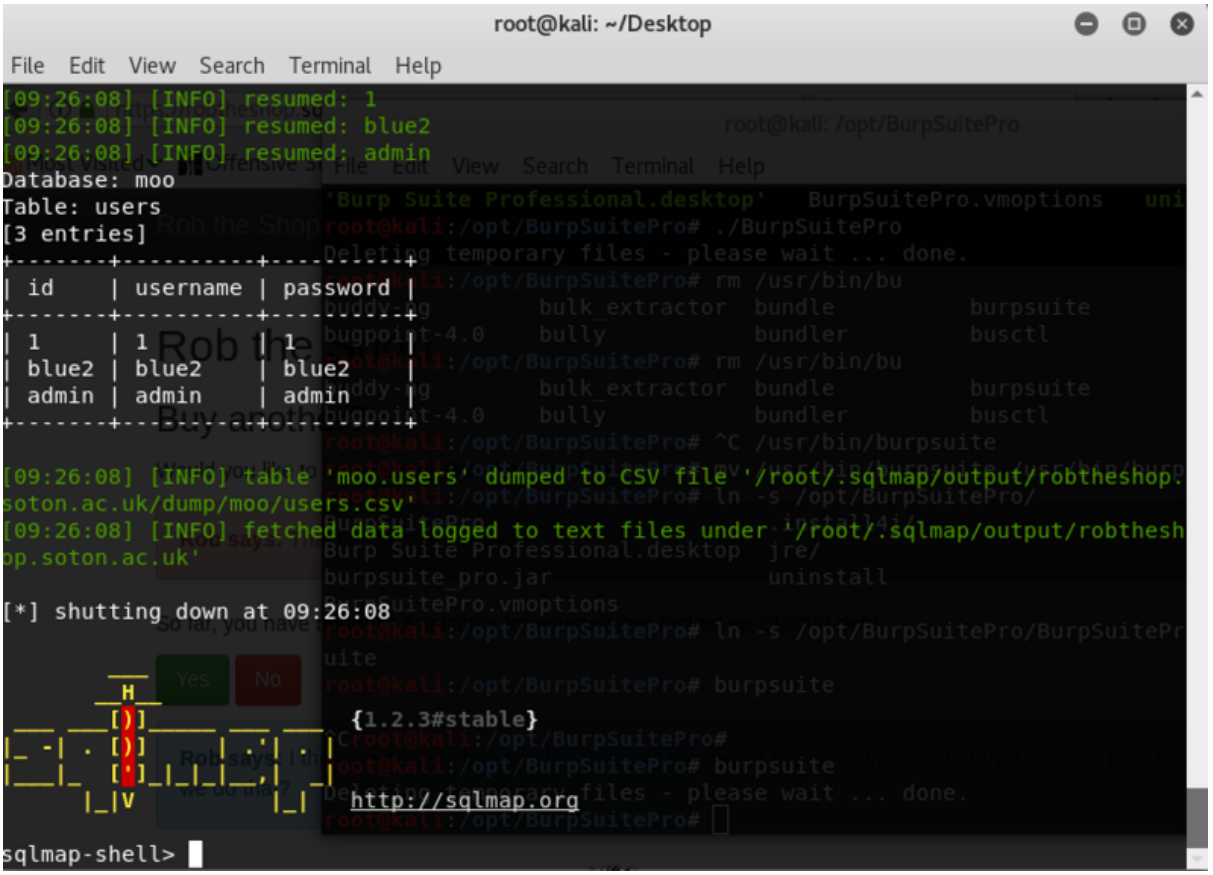
Inputs	**Query**	**Result**
Username = Oli Password = something	SELECT * FROM users WHERE username='Oli' AND password='something'	User logged in if username and password match
Username = Oli Password = something' OR 1='1	SELECT * FROM users WHERE username='Oli' AND password='something' OR 1='1'	User will be logged in as Oli because the OR binds tighter and 1='1' evaluates to true
Username = Oli' -- Password = something	SELECT * FROM users WHERE username='Oli' -- ' AND password='something'	User will be logged in as Oli because everything after the username gets commented through using the double dash

- How do you find an SQL injection vulnerability?
    - Find any inputs that look related to the database
    - Think about how the queries are being constructed

    - Try putting in special characters
    - These will break the query
    - Can you get the application to break?
    - Can you get the query to do something else?

- How can you fix this vulnerability?
    - The best solution is to use prepared statements
    - Prepared statements have placeholders in them, which get substituted by the user input
    - By using prepared statements you are telling the database what shape of query to expect
    - Therefore, it won't allow any inputs that alter the functionality of the query

- sqlmap is a tool which given a single SQL injection can extract the entire database out for you
    - So it only takes 1 SQL injection vulnerability to expose the whole database



Authorisation Bypass

- Many web applications have a restricted area that is only accessible for some users
    - E.g. an admin interface

- These require authorisation to access but can we bypass this authorisation somehow?

- We can try to just guess a URL
    - Where might the admin pages be located?
    - Maybe it's at /admin?
    - We look at the URLs found so far and then we can maybe spot a pattern and thus guess the right URL
- If we are at http://www.amazingshop.com/products could there be http://www.amazingshop.com/admin/products?

- We can also try to submit forms - or in general to send any sort of HTTP requests - to URLS we shouldn't be allowed to
    - Can we create a new admin if we send a POST request to admin/add?

- How to fix this vulnerability
    - For every action think about the authorisation they require
    - On the server side check if the user is allowed to perform this action

Cross-Site Scripting (XSS)

- XSS is when the user can not only provide text but also code for some input which is used as an output somewhere else
- Thus, it will execute the HTML/JavaScript code provided by the user
- We are basically "reprogramming the page"

- There are two main types of XSS
    - Reflected XSS: The input is displayed back immediately
    - Stored XSS: The input is stored in a database and displayed back later

- For example, suppose that a user can set a personal greeting on the profile page
    - The user enters "My name is Oli" and the page will display this greeting to every user that visits this profile

```
<html>
  <head></head>
▼<body>
    <h1>A simple website</h1>
  ▶<form action method="post">…</form>
…   <h3>Hello Oli</h3> == $0
  </body>
▶<div>…</div>
</html>
```

    - That's fine, so where is the issue?
    - Suppose the user enters "My name is <marquee>Oli<marquee/>"

```
<html>
  <head></head>
▼<body>
    <h1>A simple website</h1>
  ▶<form action method="post">…</form>
  ▼<h3>
      "Hello "
...     <marquee>Oli</marquee> == $0
    </h3>
  </body>
▶<div>…</div>
</html>
```

    - Now it will show “Oli” with the marquee effect
    - We could put any code there (including JavaScript) and it would get executed!

- How to find XSS vulnerabilities?
    - Find any place where user input is displayed, either immediately or later
    - Try inserting HTML and JavaScript and see if it makes it back through
    - Use the XSS cheat sheet https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

- How to prevent XSS vulnerabilities?
    - We can use libraries that sanitise the user input
    - For example, the < and > brackets can be replaced by the HTML entities < and >

A simple website

Enter your name: Submit

Hello <marquee>Oli</marquee>

```
<h3>Hello &lt;marquee&gt;Oli&lt;/marquee&gt;</h3>
```

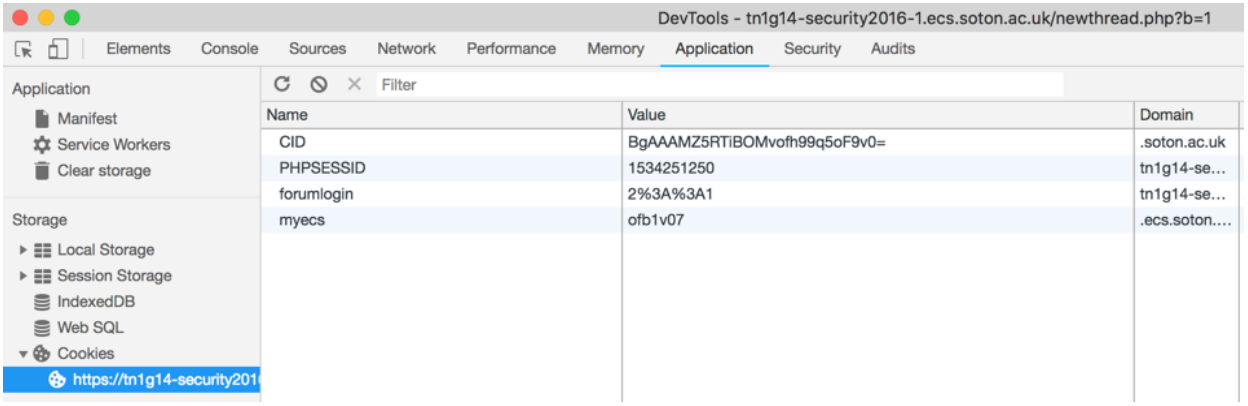
Insecure Cookies

- Many applications use cookies to remember that you were logged in
- It is important to implement this functionality the right way otherwise your application is vulnerable

- For example, an application stores the username of the user currently logged in in the cookies
    - Cookies are stored in your browser
    - You can change everything that is stored in the cookies

    - Just change the username to something else and you will be logged in as someone else
    - Obviously, this is a huge vulnerability!

- How can you find insecure cookies?
    - Check what cookies are being set
    - What values do they have



    - How do they change with different accounts?
    - What happens if you change them?

- Sometimes cookies contain encoded information
    - Can you decode it?
    - Sometimes it's just a base64 encoded string
    - Encoding does not make it more secure
    - You can decode, change the value, encode it and change the value of the cookie

- How to fix this vulnerability
    - Make sure that the user cannot make any damage or cause unexpected behaviour if they change the cookies
    - Do not use any information in the cookies without validating it again on the server

Insecure Sessions / Session Vulnerabilities

- This related to the insecure cookies vulnerability
- How do sessions work?
    - When you log in with your username and password you are given a session
    - It is this session which keeps you identified to the server
- If someone can steal your session, they become you!
- Session vulnerabilities arise when sessions are possible to steal or capture

- How to find this vulnerability?
    - Figure out how the session is stored

    - Usually this is done using a cookie
    - What happens when you log out and back in?
    - Is the session cookie always the same?
    - Can you guess the value of the session cookie?
    - Are there any patterns to the sessions

- How to fix this vulnerability
    - Use a randomly generated session ID to identify users
    - There shouldn't be any patterns or anything else that enables you to guess what the session ID of another user is

Insecure File Upload

- What is it: Insecure upload functionality
    - You can upload something you shouldn't be allowed to

- Why is it bad: You can upload arbitrary content to the website
    - Can set up your own phishing site on that domain
    - Can execute php on the server

- How can you find it?
    - Look for any upload forms and give them what they are not expecting
- In particular, check the following
- Filename
    - Does it change the filename that you give it?
- MIME types
    - These are sent by the browser so you can easily manipulate them
- File extensions
    - Try double extensions
    - Try archives like zip files
- Content
    - If it expects an image embed something else
    - Manipulate the EXIF headers, e.g. change the Comment with exiftool

```
root@kali:~/Downloads# exiftool -Comment="<?php phpinfo(); ?>" rob3.jpg
    1 image files updated
root@kali:~/Downloads# exiftool rob3.jpg
ExifTool Version Number         : 11.70
File Name                       : rob3.jpg
Directory                       : .
File Size                       : 199 kB
File Modification Date/Time     : 2019:10:17 09:01:55-04:00
File Access Date/Time           : 2019:10:17 09:01:55-04:00
File Inode Change Date/Time     : 2019:10:17 09:01:55-04:00
File Permissions                : rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : None
X Resolution                    : 1
Y Resolution                    : 1
Exif Byte Order                 : Little-endian (Intel, II)
Software                        : Google
Comment                         : <?php phpinfo(); ?>
Image Width                     : 2414
Image Height                    : 1810
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 2414x1810
Megapixels                      : 4.4
```

- Extensions are important!
    - If you want to be able to upload a PHP script for the server to run, it needs to end in .php
    - If you upload a php file as a jpg it won't get sent to the PHP processor to execute it. It's just an invalid image at this point
    - Unless you have the ability to run things on the server in another way – then you can upload whatever you want wherever you want and then require/include/execute it

- For example, take the upload functionality for the user's avatar
    - Let's upload a PHP script!
- If it just checks the MIME type let's fake that as well
    - MIME type is sent by the browser so we can send anything

- How to fix it?
    - Use a whitelist not a blacklist
    - Check the files in as many ways as possible

Cross-Site Request Forgery (CSRF)

- This means tricking the browser into doing something you didn't intend to do

    - Actions are performed without verification or awareness of the user
- For example, you are logged in to website.com and click on a link that goes to website.com/index.php?action=deletethread&threadid=1
    - Your browser sends a GET request to this URL
    - The thread will be deleted!

- For example, here is the admin panel for adding a new user


```
<h2>Add User</h2>

<form action="adduser.php" method="post">

<p>Username:<br>
<input type="text" name="username">
</p>

<p>Password:
<br>
<input type="text" name="password">
</p>

<input type="Submit" value="Add User">

</form>
```

- This is a very simple form that sends a POST request with username and password to admin/adduser.php
- Now here is the page from the attacker


```
<p style="text-align: center">
<img src="https://www.pets4homes.co.uk/images/articles/4295/l
arge/early-neutering-of-kittens-pros-and-cons-598ddb68021a9.j
pg" style="width: 100%; height: auto;">
</p>
<form action="http://goodsite.com/admin/adduser.php" method="
post">
<input type="hidden" name="username" value="BadUser">
<input type="hidden" name="password" value="NaughtyPassword">
<p style="text-align: center">
<input type="submit" value="Show More Kittens">
</p>
</form>
```

- Isn't this cat cute? 😺
    - I'm sure you want to see more, so you click on "Show More Kittens"
- What you don't realise is that when you press the button you submit a POST request to admin/adduser.php
    - There are 2 hidden fields with username and password, which will be sent in the POST request
    - If you are logged in it will create the user!

- How do you find this vulnerability?
    - Are there any GET requests that manipulate data? GET requests should only be used to show data
    - Are there any forms that do not have a CSRF token? (more on this below)

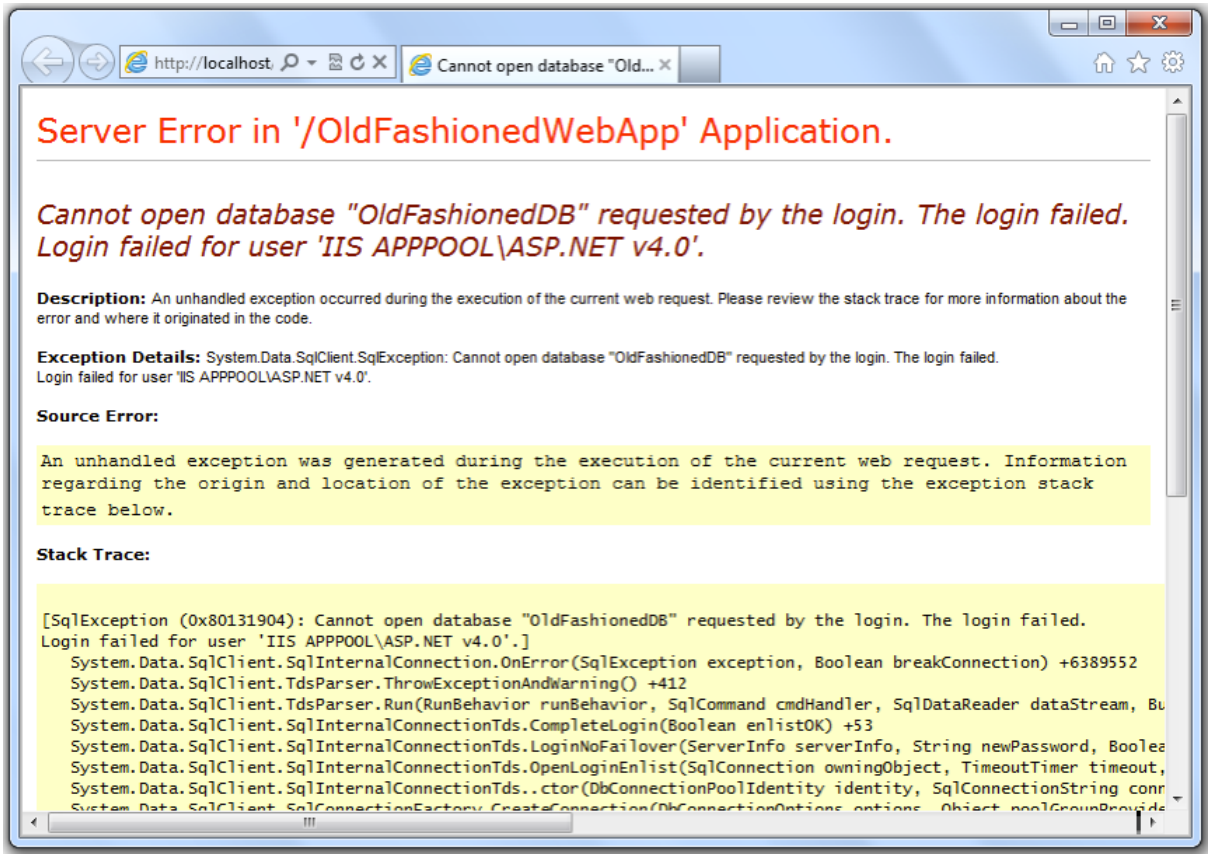
- How do you fix this vulnerability?
    - If you find any GET requests that manipulate data, change them to POST/PUT/DELETE
    - Make sure to add a CSRF token to all actions that manipulate data
- A CSRF token is a random token that is generated by the server
    - When the page loads a random one-time CSRF token is generated
    - It should not be possible to guess what the CSRF token of a user is
- Every request that manipulates data should contain the CSRF token
    - We will include the CSRF token as a hidden field in every form
    - Now every form is different because it will have a different CSRF token
    - When the server receives the request it will validate the token
    - Only if the CSRF token is correct we will allow the action
- This way an attacker cannot just send you a link or make you submit a form to perform an action on behalf of you
    - Here is a secured version of the admin/adduser.php form
    - CSRF token is called nonce here

```
<h2>Add User</h2>

<form action="adduser.php" method="post">

<p>Username:<br>
<input type="text" name="username">
</p>

<p>Password:
<br>
<input type="text" name="password">
</p>

<input type="hidden" name="nonce" value="5d41402abc4b2a76b9719d911017c592">

<input type="Submit" value="Add User">

</form>
```

```php
if($_POST['nonce'] != $_SESSION['nonce']) {
  error("CSRF attempt blocked");
}
```

Internal Information / Information Leakage

- Have you ever visited some website and seen something like this before?



- No, I don't mean Internet Explorer, I mean the error message on that page
- It exposes a lot of internal information about the application
- The standard user should not see any of this information!

- It first it does not look like a big deal when you expose all this information
- The attacker needs information
  - The more information, the easier to attack
  - If the attacker knows the version of the software you are running they can look up known vulnerabilities for it
- Maybe the error message includes your database credentials as well…

- How to fix this?
  - Make sure that all errors are handled
  - Don't show any error messages (with internal information) or stack traces to normal users

    - Instead show very generic error messages

Parameter Manipulation

- Parameter manipulation means changing the values of any parameters (user input) used in the application and thus causing it to do something unexpected
    - This works when inputs are not (properly) validated on the server
- It is not enough to only have client-side validation
    - This includes client-side scripts, form controls etc
    - An example is using a combo box to limit choices
    - An attacker can edit the contents of the combo box!
    - TLDR: You can never trust the client!

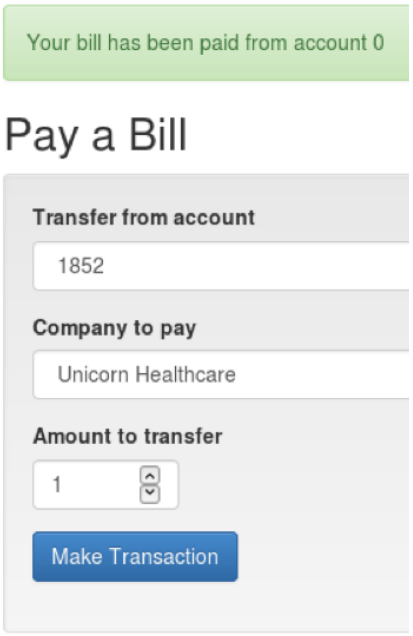
- How to find this vulnerability
    - Work through all form fields
    - Try manipulating all form elements and parameters and see if you can pass data they shouldn't allow
    - Think about what validation is or should be done
- Some things you can try out…
    - Different data types - text instead of a number
    - Decimals instead of a number
    - Go out of range - negatives, positives, zero
    - Leave out a parameter
    - Add a new parameter
    - Overflow the input with an extremely long input
    - SQL injection
    - Commands - JavaScript, PHP, Bash etc
    - Anything else that isn't expected

Application Logic

- This one is related to parameter manipulation
- It is about breaking the logic of the application or finding logic bugs
    - Attackers cause unexpected behaviour in your application
    - They get where they shouldn't be

- Example: Creating an unpublished blog post in the admin panel
    - Can a normal user see it on the home page?
    - Can a normal user see it when searching for a word that is in the text?
    - Can a normal user see it when showing all posts for a certain category?

- Example: Paying a bill




```
$account = getAccount($data['Transaction']['from']);
$to = getBillAccount($data['Transaction']['reference']);
$amount = $data['Transaction']['amount'];

If(!$account) {
        error('Invalid account to pay from');
}

if(!$to) {
        error('This is not a valid company');
}

if($account->balance > $amount) {
        $account->transfer($to,$amount);
} else {
        error('You do not have the necessary funds');
}
```

- The problem here is that the account from which the money will be taken comes from the form
    - An attacker can change the "from" account to any account they like
    - The "from" account shouldn't come from the form but be based on the user that is currently logged in (session)

- What to look for
    - Where user input is processed not just queried/stored
    - Multi-step processes
    - Any parts of the application logic which the user can manipulate
    - Is there any application logic which the users can affect through user input?

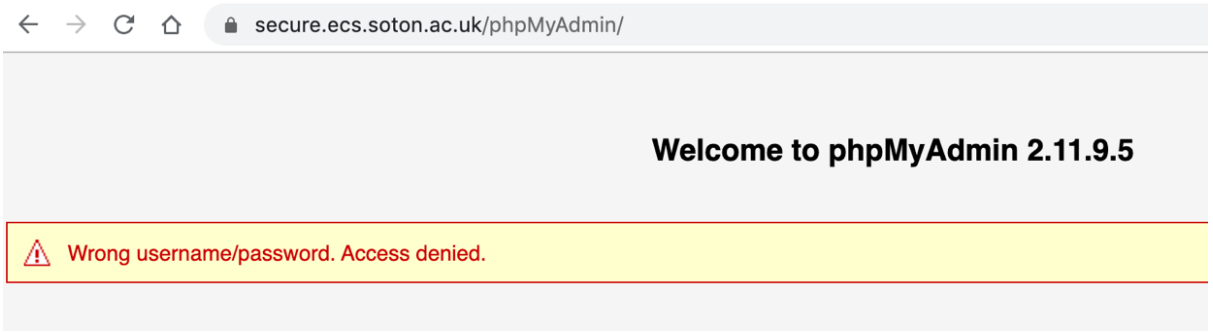
- How to fix it
    - Basically don't introduce bugs into your application, easy innit 🤷‍♂️
    - Jokes aside, that's one of the reasons why you should test your application
    - Make sure you got the logic right
    - Use the current session to identify the user that is currently logged in and do not use any (hidden) form fields for this purpose!

Out of Date Software

- This one is actually so easy to fix but yet responsible for so many compromises
- It's so easy to set up a website, install some software, get it running and everyone is happy
    - Until it's still running (the same versions) 3 years later
    - And the software is full of security holes
- There are multiple layers that can be out of date

    - The core of the web application
    - Libraries
    - Operating system

- What you are looking for
    - Is the software out of date?
    - Is any components or underlying software out of date?
    - Signs of what software the site is using
    - Signs of the version they are using



    - Any vulnerabilities in that version that can be exploited

- Once an attacker knows what software and version you are running they can use the national vulnerability database to find known vulnerabilities
    - https://nvd.nist.gov/

- What to think about
    - How can you get updates to deprecated software?
    - Don't make it obvious which software and version you are running

File Inclusion

- File inclusion is when you can include files on a site that you shouldn't be able to access
- That's a bit vague, let me illustrate this with an example
    - A page includes (shows) a file based on a query parameter - the file name comes from that parameter
    - Say you go to /help?topic=cybersecurity so the page loads the file cybersecurity (located in the current directory) and displays its contents
    - However, what happens if you try to include any files outside of the current directory?
    - So if you enter a topic like ../../allpasswords will it show this file as well?
    - If it does you just found a file inclusion vulnerability!

- How to find this vulnerability
    - Are there any places in the application that include files?
    - Is this done based on user input?

    - Can you only include the files that you want to be included?
    - Can you change what is included?

- Sometimes you might need to encode or double encode the parameter to exploit this vulnerability
    - Poor security mechanisms only decode once

- How to fix this vulnerability
    - Make sure there is absolutely no way of including files outside of the current directory
    - Do not allow any slashes in the parameter because a file can't have slashes in its name and you don't want to allow directory traversal!

Open Redirects

- An open redirect is when you can get the application to redirect you to another third party website
    - And that is the website from the attacker

- For example, after a login the user gets redirected to /postlogin?destination=home
    - The query parameter home defines the destination of the redirect, thus it will redirect the user to /home
    - What if you put /postlogin?destination=//evilwebsite.com (possibly encoded or double encoded)
    - If the website redirects the user to evilwebsite.com then you have found an open redirect vulnerability!

- How to discover this vulnerability
    - Find any redirects, you can identify them by 3xx status codes
    - To find a redirect with input, look at the request
    - Can these URLs be redirected to other pages?

- How to fix this vulnerability
    - It's best to create a white list with allowed destinations
    - If the destination doesn't match, don't redirect or redirect to a default location
    - If you can't use a white list then don't allow any slashes in the destination (think of encodings and double encodings as well)

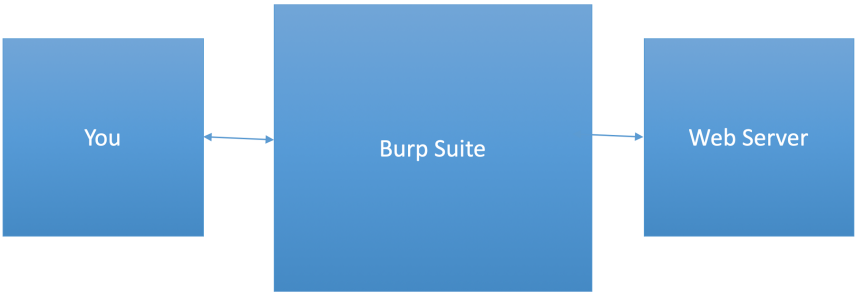
Brute Force

- Are there any actions that can cause damage when repeated many times?
- Example: Login form
    - Just spam 1000s of requests to find out a user's password

- How to fix it
    - Use recaptcha after a certain amount of failed login attempts
    - Limit the amount of requests a user can send

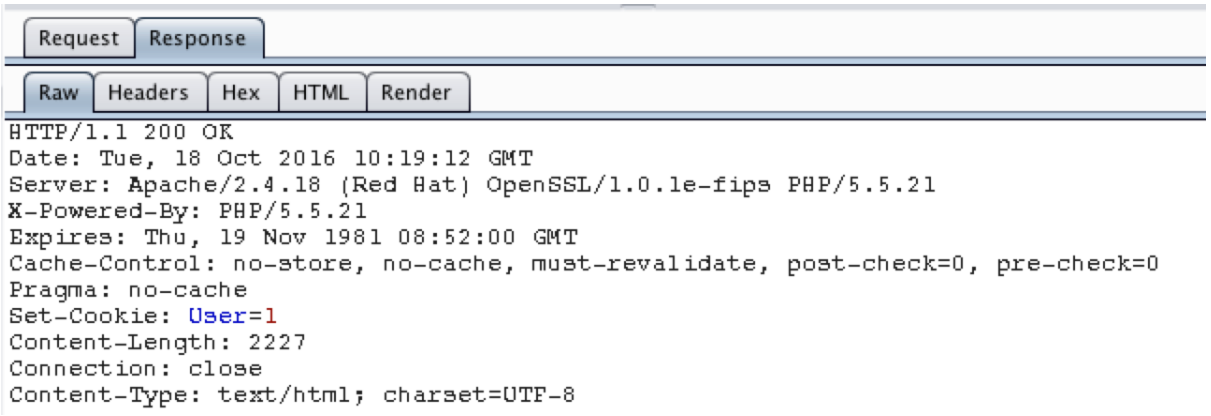
Tools & Techniques

- Let's look at some tools that make it easier for us to discover vulnerabilities
- The first tool we are going to look at is Burp Suite
    - It is an intercepting proxy
    - All your traffic goes through it and you can manipulate it
    - It collects information and builds a site map
    - You can check what you have done so far



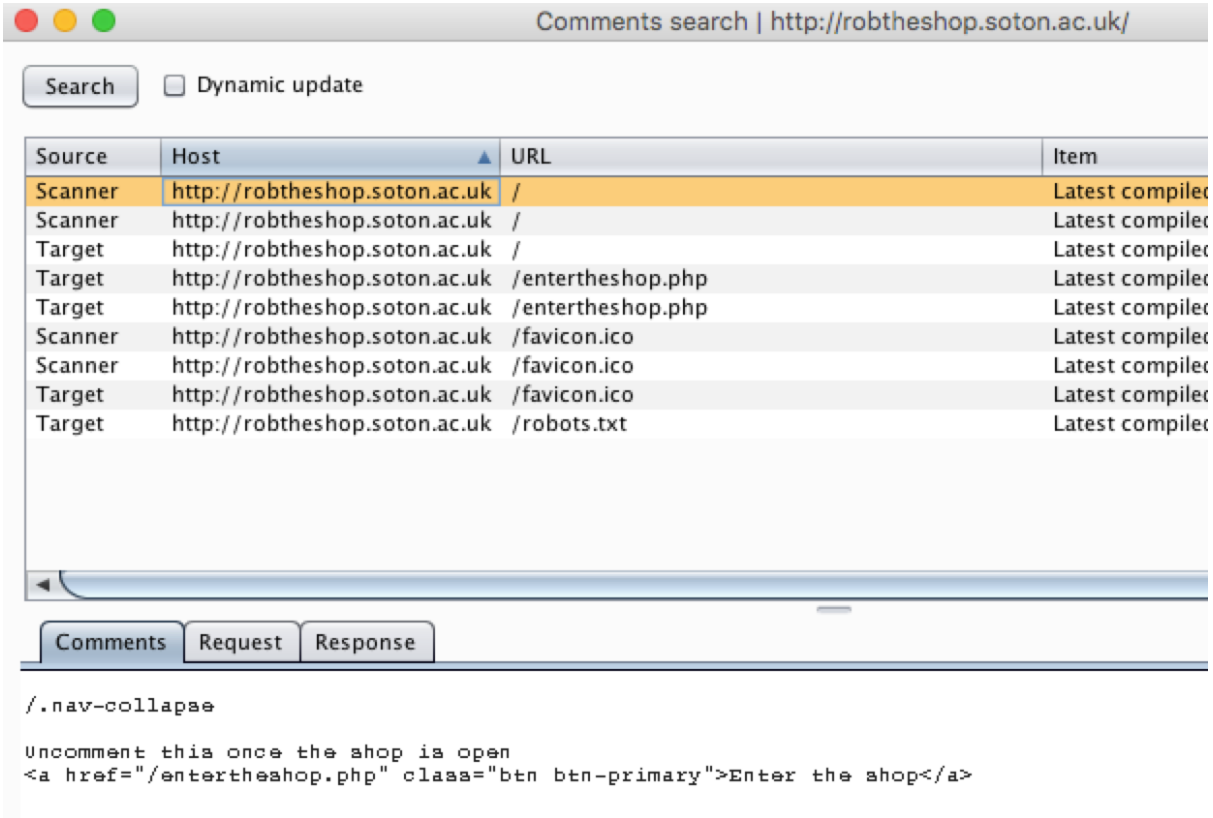
- In order for this to work you need to set up your browser correctly
    - Your browser needs to send all traffic to Burp
    - For HTTPS to work you should add the certificate from http://burp/cert

- Using Burp we will try to find vulnerabilities in the online shop robtheshop
- This is what it will look like when Burp logs a response



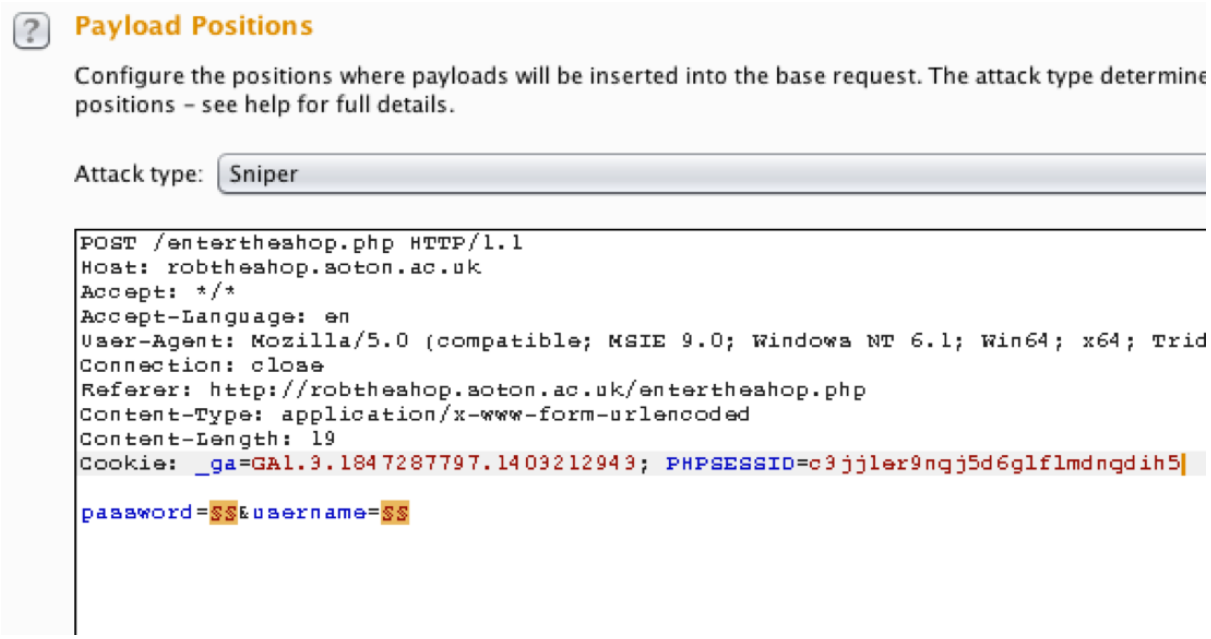
- From this we can already spot that it sets a cookie User=1, which might be a vulnerability

- Burp can also show us URLs that are commented in the HTML and thus not rendered in the browser

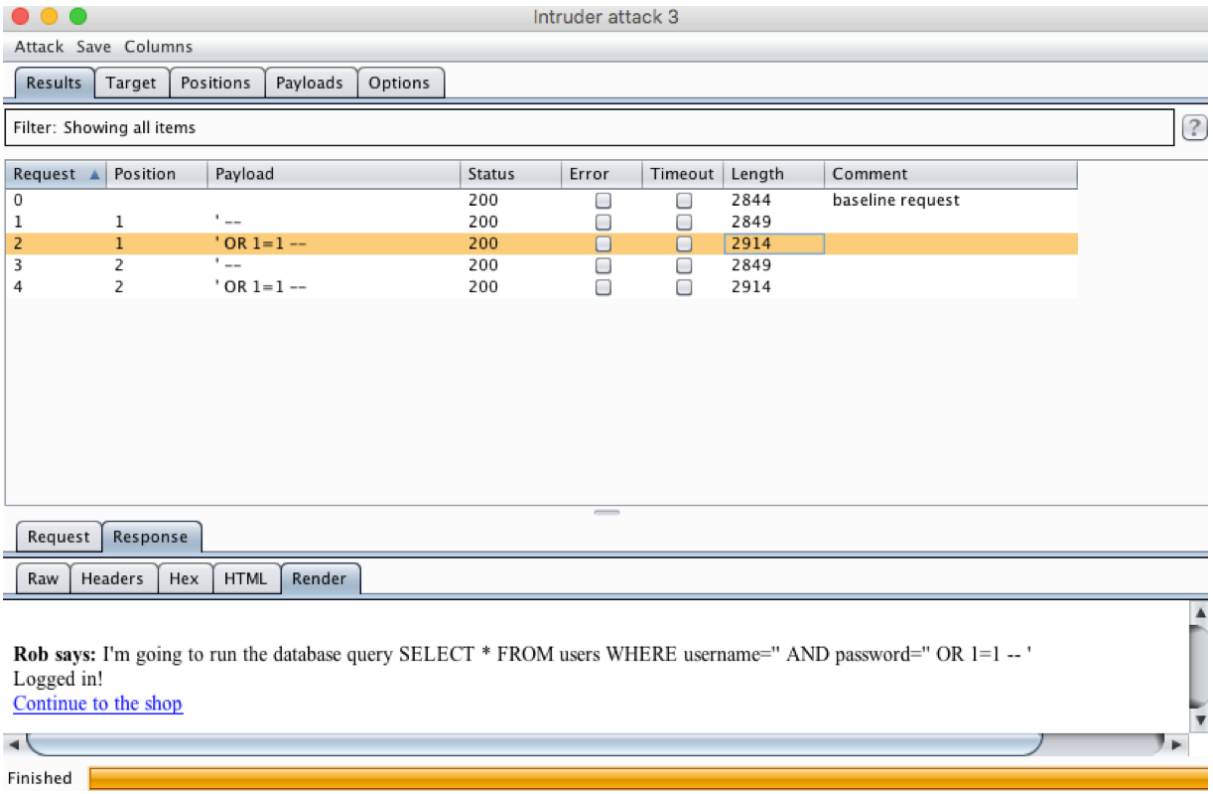


- We have just found that there exists an entertheshop.php

- We might come across a login so we can try to perform SQL injection
- We find the request that sends the username and password to the server and manipulate it in Burp

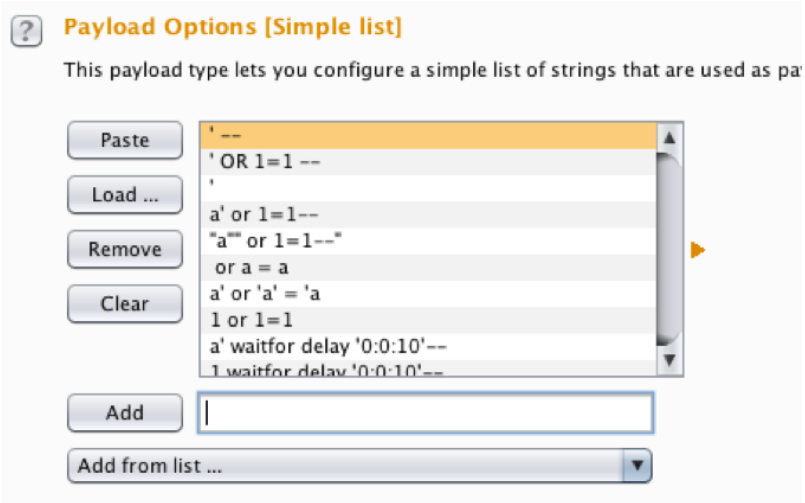


- Using the Intruder and a list of dodgy inputs we try different values for username and password until we find that there is a vulnerability

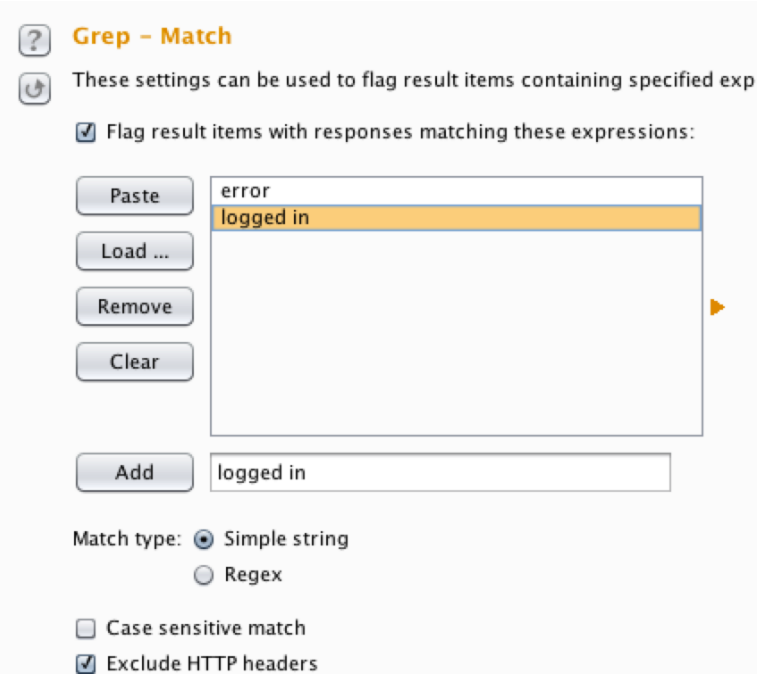


- The response preview shows that we are logged in

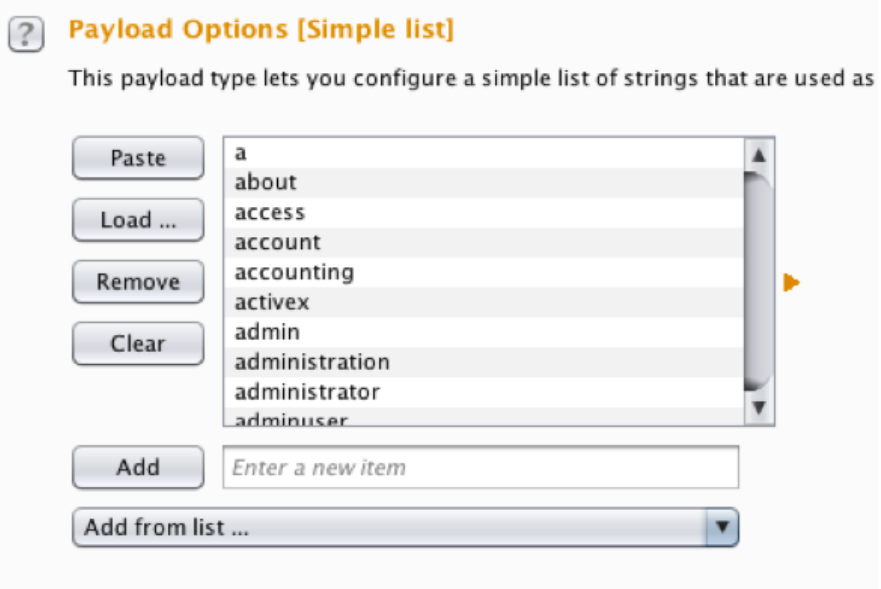
- Trying out different values from a list called fuzzing
    - We use lists of dodgy inputs that have been precompiled by people
    - We can use this for SQL injection but also for XSS
    - We can use other lists to guess URLS



- In order to find out when we have succeeded we can look at the response length or use a filter like grep to match with phrases like “logged in”



- As mentioned above we can use other lists to guess URLS
- This is helpful to find any leftover files or authorisation bypass vulnerabilities



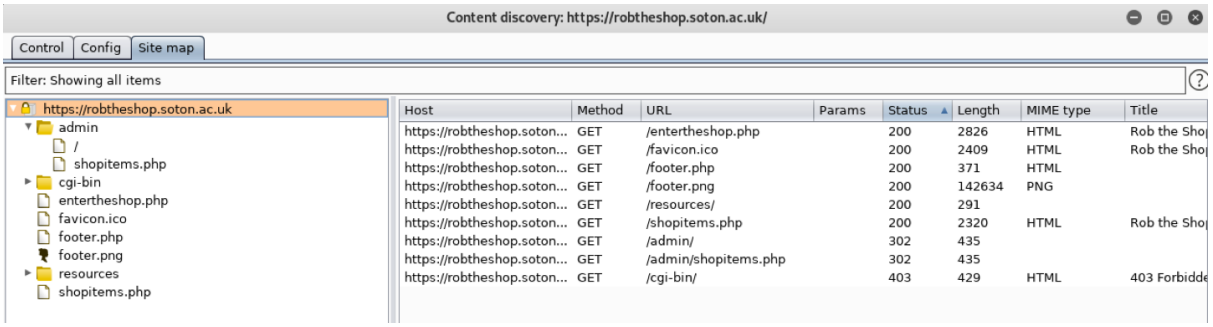
- To find the auth bypass, we could just use this big list, however, let's enter the realms of informed guessing
    - We know that there is an admin folder
    - We know that all the file names end in .php
    - We know that the site uses the term "item" to refer to things in the shop from what we have seen already
- When we put all those things together, we can drastically reduce the amount of stuff we need to guess/test out
- While we can do this manually, there is also a tool called "dirb" that can do the work for us

```
root@kali:~# dirb https://robtheshop.soton.ac.uk

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Thu Oct 12 05:52:43 2017
URL_BASE: https://robtheshop.soton.ac.uk/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: https://robtheshop.soton.ac.uk/ ----
==> DIRECTORY: https://robtheshop.soton.ac.uk/admin/
```

- As you can see above, it found the admin directory
- Now we can use dirb to find pages in this directory

```
root@kali:~# dirb https://robtheshop.soton.ac.uk/admin -X .php

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Thu Oct 12 05:58:09 2017
URL_BASE: https://robtheshop.soton.ac.uk/admin/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
EXTENSIONS_LIST: (.php) | (.php) [NUM = 1]

-----------------

GENERATED WORDS: 4612

---- Scanning URL: https://robtheshop.soton.ac.uk/admin/ ----
+ https://robtheshop.soton.ac.uk/admin/index.php (CODE:302|SIZE:0)
+ https://robtheshop.soton.ac.uk/admin/items.php (CODE:302|SIZE:0)
```

- And we have found that /admin/index.php and /admin/items.php exists
- Alternatively, you can find the same using content discovery in Burp



- When using word lists for fuzzing be careful that you don't send too many requests!
- This might trigger intrusion detection or firewalls

- How do we keep track of where we can enter data?
- We can use filters in Burp to only show parameterised requests

- Let's take a closer look at the quantity parameter on the buy page
- What happens if we specify 0 as the quantity? What happens if we specify a negative amount?

- When trying to find XSS we try loads of special characters and check how the website handles it
- Does everything come back as we input it?

Penetration Testing

- How do you approach penetration testing systematically?
- Define the scope clearly
    - You don't want to access machines you shouldn't be
    - You don't want to miss out systems you should test
- Gather information
    - Application logic: Where is some processing involved that could lead to a vulnerability?
    - URL structure: Are there some frameworks being used that introduce a certain pattern

    - Inputs and outputs: Are there some hidden inputs? Are all inputs validated?
    - Error handling: How are errors handled? Do you get any logs?
    - Sessions and cookies: How do they work? What happens when we change cookies
- Identify vulnerabilities
- Exploit: eliminate false positives
- Post-Exploit: what can you do now?

- Some ideas for your thought process
    1. Be systematic: Make a list of all possible inputs so you make sure that you don't miss anything
    2. Be logical: What is the code behind the logic? If you were the developer, what code would you write to implement this logic?
    3. Map out what you're doing: Create a spreadsheet to show the structure of the page
    4. Use tools to help you, e.g. Burp Suite
    5. Be the anti user: Test user inputs that are usually not expected, e.g. a negative number, a string when there is a number expected etc.
    6. Intercept and manipulate: Use tools like burp suite to change the request
    7. Cheat: Use cheat sheets
    8. Don't give up
- We will look at these steps in detail

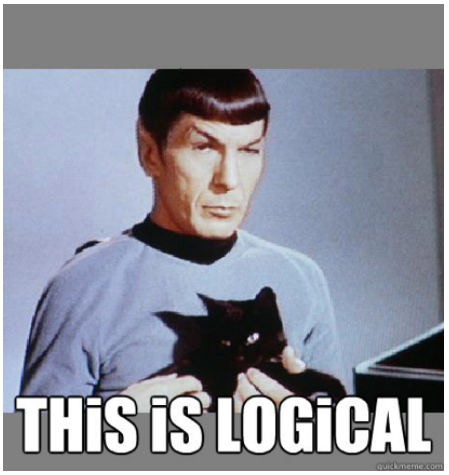
1. Be systematic

- Know what you are looking for
    - There is a list of top vulnerabilities
    - Check if the application you are testing has any of those

- Out of date software
    - Can we find out what software is running?
    - Can we find vulnerabilities for that software?
    - Can we find exploit code for that software?
- Information Leakage
    - Can we find anything that's hidden on pages?
    - What are the error messages like?
    - What are the unexpected error messages like?
- Session Vulnerabilities
    - What are the sessions like?
    - Are they using their own system or the language built in sessions?
    - Some people think they can make something better
    - They usually fail
- SQL Injection
    - What parts of the site use the database?
    - What inputs do I have into those parts?
- Cross-site Scripting
    - Where can I input data into the site?

    - Where am I inputting data into the site without realizing?
- Cross-site Request Forgery
    - GET requests aren't supposed to change state
    - Can you find any GET requests that do?
    - Do the forms seem protected?
- Authentication Bypass & Authorisation Bypass
    - What should I be able to do as a guest/user/admin?
    - What can I actually do?
    - Is the process protected the whole way through?
- File Upload Vulnerabilities
    - Upload ALL THE FILES!
    - Fake fake fake fake!
    - Fake the content, fake the type, fake the filename, fake the headers, fake the properties – fake it all!
    - And then fake some more!

2. **Be Logical**

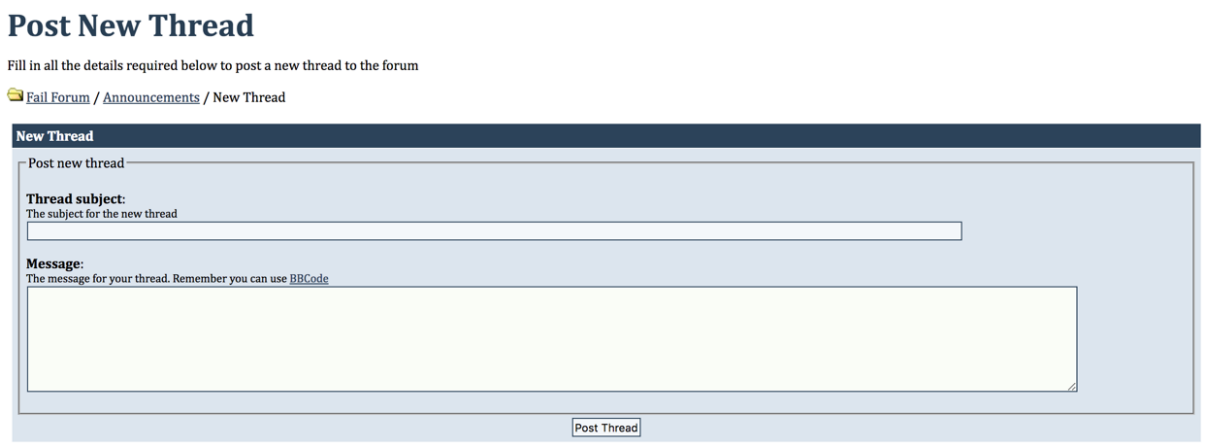


- Continue from being systematic
- Think about
    - What is the code doing?
    - What interactions are taking place?
    - Where could the vulnerabilities be?
    - What might we have in the database?
    - Which parts take user input?
    - Which parts of the size does this occur on?

3. **Map out what you're doing**

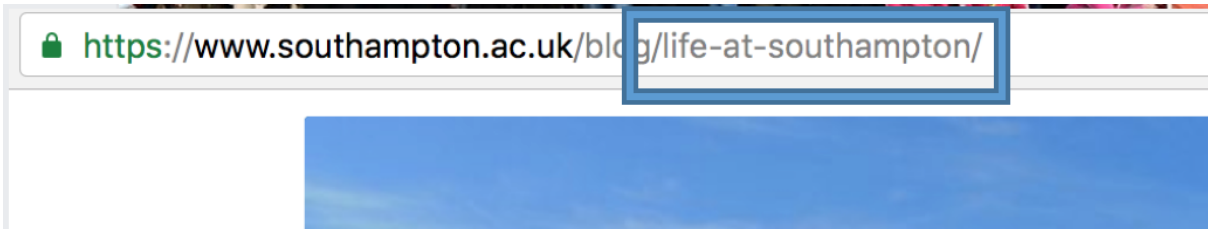
- You are testing a system you have never seen before, it's like going to a new city
- Before diving in trying to find and exploit things, you want to map out the system
- Use the tools to help you do this
    - But sometimes some pen and paper is good to

- What areas of the website are there?
- How does each area link to each other?
- What forms do they have?
    - What inputs are there?

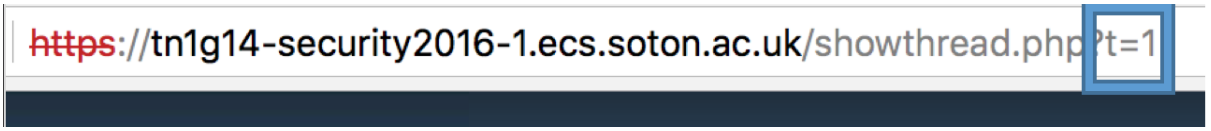



```
▼<form id="newthread" name="newthread" action="https://tn1g14-security2016-1.ecs.soton.ac.uk/newthread.php?do=post" method="post"> == $0
  ▼<fieldset id="post" class="fieldset">
      <legend>Post new thread</legend>
      <input id="board_id" name="board_id" type="hidden" value="1">
    ▶<p>…</p>
    ▼<p>
        <strong>Message</strong>
        ":"
        <br>
      ▶<small>…</small>
        <br>
        <textarea id="post_message" name="post_message" rows="40" cols="10"></textarea>
      </p>
    </fieldset>
  ▶<div class="center">…</div>
</form>
```

- How are they accessed?
    - What URL parameters are there?



    - What GET parameters are there?



- What are the different types of pages?
    - Static pages - they always remain the same
    - Dynamic pages - they change based on logic and/or user inputs or some other application state
    - Logic - any page where the input is processed

- Write it down in a structured way!

Key: S (SQL Injection), X (XSS), M (Manipulation), ? (Present), V (Vulnerable), E (Expected), A (Actual)

URL	GET Parameters	POST Parameters	Authorisation	Features	Vulnerabilities & Comments
Parameter / S / X / M	Parameter / S / X / M	Parameter / S / X / M	Role / E / A	Feature / ? / V	
				DB Read	
				DB Write	
				Logic	
				CSRF	
				Upload	
				Email	
				SetCookie	

URL	GET Parameters	POST Parameters	Authorisation	Features	Vulnerabilities & Comments
/users/view/<id>	Parameter / S / X / M	Parameter / S / X / M	Role / E / A	Feature / ? / V	Parameter Manipulation + Internal Information: Error produced when type parameter is invalid with stack trace
Parameter / S / X / M		name	Admin / X / X	DB Read	
<id>		type / X	User / X	DB Write / X	Authorisation Bypass: User can edit profile.
			Guest	Logic	
				CSRF / X / X	Vulnerable to CSRF and file upload
				Upload / X / X	
				Email	
				SetCookie	

- Write down each page and tag them with what we might find there
  - For instance, use tags P for URL parameters, R for reflected XSS and D for database and so on…
  - We think that we might find those 3 vulnerabilities on the accounts page so we write down
    View (/accounts/view/<id>) [P] [R] [D]
  - Once we start testing we go through all the pages and tags to see if we can find any vulnerabilities
  - We can also put this in a spreadsheet

URL	Form ID	Name	Type	Example	Database	Reflected	SQL Injection	Reflected XSS	Stored XSS
/transactions/transfer	1	TransactionTransferForm	_method	hidden	POST	No	No	n/a	n/a	n/a
/transactions/transfer	1	TransactionFrom	data[Transaction][from]	select	3307	Yes	Yes	No	No	No
/transactions/transfer	1	TransactionTo	data[Transaction][to]	number	1	Yes	Yes	No	No	No
/transactions/transfer	1	TransactionReference	data[Transaction][reference]	text	My reference	Yes	Yes	No	No	No
/transactions/transfer	1	TransactionAmount	data[Transaction][amount]	number	10	Yes	Yes	No	No	No

URL	Name	Type	Example	Database	Reflected	SQL Injection	XSS	Invalid Data
/transactions/index/<id>	filter	number	1	Yes	Yes	Unknown	Unknown	Unknown

URL	Name	Type	Example	Database	Reflected	SQL Injection	XSS	Invalid Data
/transactions/index/<id>	id	number	1	Yes	Yes	Unknown	Unknown	Unknown

Form ID	URL	Method	Action
1	/transactions/transfer	POST	/transactions/transfer

Table	Field	Type	Example
Transaction	from	number	1
Transaction	to	number	1
Transaction	reference	text	Hello
Transaction	amount	number	1

4. **Use tools to help you**

- There are lots of great tools out there
- Find the ones that work best for you
    - There is no perfect tool
    - It depends on how you work and what you want to do
- One such tool is Burp Suite

5. **Be the anti-user**

- You've been logical, now it's time to be illogical!
    - See a number? Let's try a string
    - Or maybe a negative number
    - Or a number that's too large for the database
- See a value? Let's leave it empty
    - Or maybe lose it all together… Oops
- See a field? Let's overflow it
    - Spam spam spam spam spam
- It's hidden?
    - So what!

6. **Intercept and manipulate**

- Use the browser developer tools for the basics
- Use Burp Suite for the more advanced manipulation
    - Turn on intercepting and see what is being sent and what is being received
    - Then mess with it!

7. **Cheat**

- People have done a lot of this work before
- There are cheat sheets out there to make your life easier

    - Particularly when it comes to XSS
- XSS isn't always as simple as <script>alert("hello")</script>
- Check the next step for an example
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

8. Don't give up

- Be exhaustive – it's not always that simple!
    - Sometimes you have to be clever
- Users put the URL as
    - http://blah.png" onMouseOver="alert(blah);
- Run through htmlspecialchars, that's still
    - http://blah.png" onMouseOver="alert(blah);
- So when it bungs the url inside the <img> tag, we get
    -
    -
- And when the user moves the mouse over the image they execute the javascript!

Ed's Part

Threats - range and type

- A security system is only as strong as its weakest link
    - Your are the weakest link ….. okay we already had this one before
- Perfect security doesn't exist
    - It is a compromise between usability/accessibility and security
    - You have a laptop and you want no-one to be able to access the data, so why don't you put it in a block of concrete and throw it in the sea?
- If you want to find vulnerabilities, think like an attacker
    - There are usually multiple lines of defense in a well secured system

- Some potential threats are
    - Visual surveillance: Just watch someone typing in their password
    - Wire-tapping: Install a keylogger on a system
    - Key interception: You obtain a key that is used to encrypt the data
    - Impersonation: Pretend to be someone else
    - Data duplication: You are not dealing with the original but with a copy
    - Cryptanalysis: Try to crack the encryption
    - Social engineering: Gather personal information about the victim and use it to your advantage
    - Physical security: If you can just walk in, then what's the point of cyber security?
    - Malware: Malicious software
    - Combinations of the above

- Who are potential attackers?
    - Criminal
    - Competitor
    - Hacker (or cracker)
    - Government
    - Terrorist
    - Ethical hacker: You should get written permission first
    - Former employee: Someone who has been fired
    - Employee: Someone who is to be fired soon
    - Customer
    - Contractor: Edward Snowden was a contractor for the NSA
    - Whistleblower
    - Or maybe just everybody?
- Who represents the greatest threat to your security?
    - Your own staff!

    - Why? Because they have been in your company for ages (at least some of them)!
    - There might be poor policies, poor training, complacency and/or incompetence (or stupidity)
    - Security products (like anti virus) often appear to offer an illusion of cyber security

- What systems can be attacked?

1. **Stand-alone computers (no network connection)**

- A system that is not connected to any network
    - Is there physical security?
    - Is it password protected?
    - Can you remove data?
    - Where does it get its software upgrades from?
- How do you ensure physical security?
    - Solid walls and doors (multiple?)
    - Entry via roof or underground available?
    - Use solid locks and have good key control
    - Have CCTV cameras and alarms
    - Are there security guards 24/7?
    - The same applies for backup/other locations

2. **Distributed computer (private network)**

- A system with a network connection but no connection to the outside world / internet
- The same threats as before plus
    - You can compromise a private network
    - In a lot of cases another system with internet access is introduced later, e.g. for the convenience of installing software updates.
    - People totally forget about the initial security motivation for the isolated network!
- If the system is connected to the internet, this fact makes it even more vulnerable
    - Now it's much easier for an attacker to launch the attack via the internet
    - DDOS attacks
- If the system has a website on it (website), there are even more threats
    - Can exploit vulnerabilities on the website software
    - OWASP lists more potential threats

3. **Distributed computer (internet)**

- A system with a network connection to the outside world / internet
- The same threats as before plus
    - It is now much easier for an attacker to launch an attack via the internet
    - It is now also vulnerable to (D)DOS attacks - (Distributed) Denial Of Service

4. **Server with website (internet)**

- A server with a network connection to the internet for the purpose of offering a service, e.g. a web server

- The same threats as before plus
    - Even easier for an attacker to exploit a vulnerability because the website software and web application itself are vulnerable as well
    - Common vulnerabilities were discussed in Oli's part

Malware

- What is malware?
    - It's short for malicious software
    - It has the purpose of causing unwanted effects on your computer
    - For example it disrupts computer operation, collects sensitive information (also called "spyware") and accesses private computer systems
- Malware may take many forms
    - "Virus" is the most well-known example
    - More about it below
- Often, it is designed to remain undetected
- You should protect systems using firewalls, anti-virus etc.

- Why do people make malware?
    - The purpose is taking some control
    - Usually it only takes partial control because otherwise the malware may be revealed
    - Usually for the benefit of someone else (money)
- What does malware do?
    - It may send spam to other users
    - It may steal your passwords or identity, e.g. access your bank account
    - It may store illegal content on your system
    - It may track all your activities
    - It may use your system to launch a DDOS attack

- What malware variants are there?
    - Virus
    - Worm
    - Bots
    - Trojan Horse
    - Rootkit
    - Adware
    - Scareware
    - Ransomware
    - Backdoor/trapdoor
    - Keyloggers

Trojan Horse

- What is a Trojan Horse?
  - It was used to defeat Troy defences
  - Soldiers were hiding in the Trojan Horse and attacked the city once inside
  - Hence, it is something that is not what it appears to be and contains a hidden threat
  - Malware can infect systems just like the Trojan Horse did
- Ken thompson demonstrated another Trojan horse technique in the 1970s when working on Unix
  - He changed the unix login command so that it would accept either the correct password or a master password he defined
  - He could not just write this in the code because it would be obvious in any code reviews
  - Therefore, he modified the C compiler to insert this code automatically when the code is compiled
  - Now, the Trojan cannot be detected by inspecting the source code of the login program
- Modern examples of Trojan software range from simple to complex
  - For example a phishing email that claims to have come from your bank, asking you to logon

    Secure Your Account
    From:Natwest Bank <personal@nwolb.com>
    To: Recipients <personal@nwolb.com>
    Date:Thu, 29 Aug 2013 10:59 AM
    Dear NatWest Customer,

    NatWest Bank is constantly working to increase security for all Online Banking users for the best security.

    Sign In Here To Secure Your Account

    [http://www.qastructures.com/modules/natwest.co.uk/www.nwolb.com/ibcarregister-natwst.html]

    This web site is operated by NatWest Personal Banking
    © NatWest Personal Banking United Kingdom

  - The logon screen they provide looks exactly the same as your normal bank logon screen - but of course it isn't
  - Once you enter your credentials they are sent to the attacker and thus stolen
  - It is important that the attacker redirects you back to the real banking site once they have your password so that you don't change your password once you realise that you have been conned

- Are links safe to click on?

- Go to Google and search for “ECS soton”
- You will see this result

  Electronics and Computer Science (ECS) | Electronics and ...

  *www.**ecs**.soton.ac.uk/*

- When you click on the link, does it actually go to www.ecs.soton.ac.uk directly?
- As it turns out, it doesn’t!
- It goes to this link

  http://www.google.co.uk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&ved=0CDgQFjAB&url=http%3A%2F%2Fwww.ecs.soton.ac.uk%2F&ei=5eeMUq_6H8jI0wWZ6YCIBQ&usg=[deleted]&sig2=[deleted]

- It’s a man-in-the-middle exploit that you have allowed Google to do - check the terms & conditions

Virus

- What is a virus?
  - A virus is malware that attempts to spread by making copies of itself - into other programs, data files, memory, disk drives or over a network
- It does not always have a malicious payload
- Early viruses propagated by floppy disks or programs shared on bulletin boards
  - The first reported virus was in 1982 “Elk Cloner” running on Apple II machine
  - The first PC virus was in 1986 (MSDOS)

- Now, some terminology….
  - The infection mechanism defines how a virus spreads
  - The trigger mechanism is what causes the payload to be activated - A virus may remain dormant until the trigger event
  - The payload defines what the virus does actually do - e.g. collecting information or installing software

- Today a considerable threat are email viruses
  - Virus is in the attachment
  - Recipients are usually tricked into opening them
  - An .exe file can be disguised in windows as (say) xyz.doc.exe
  - Macro viruses exploit the active content available on some applications (e.g. Microsoft Word or Excel)
- An example of an email virus
  - “I love you” (2000) was an email attachment “LOVE-LETTER-FOR-YOU.txt.vbs”
  - Not a txt file but a visual basic script!

    - .vbs is often hidden on Windows systems so users were tricked into thinking it is just a text file
    - Image files overwritten, copy of email sent to first 50 users in the Microsoft Outlook address book
    - “social engineering” – emails appear to come from friends and relations, so safe to open?
    - Damage estimated at over $5B worldwide
    - It is estimated that around 10% of all internet-connected computers (in 2000) were affected

Computer Worms

- What are computer worms and how are they different from viruses?
    - A worm can just go straight across the network, it does not have to use a mechanism like email
    - Basically, they can spread themselves without user interaction
    - A worm gains access to distant networks by exploiting vulnerabilities in services

- The first computer worm was a “proof of concept” but a code error lead to multiple infection
    - Developed in 1988 by Robert Morris Jnr - Cornell PGR student
    - Maybe 6,000 machines affected (a guess)
    - Morris convicted of felony, $10,000 fine + 3 years probation + 400 hours community work
    - Just 99 lines of code
    - Connected to another computer using vulnerabilities, run at new location, repeat
    - The vulnerability was a buffer overflow in fingerd combined with password guessing and a sendmail backdoor

- “Code Red” (July 2001) - attacks Microsoft IIS (Internet Information Services) web servers
    - About 360k infected hosts at peak (in 14 hours)
    - Exploited Vulnerability in IIS indexing software using a buffer overflow
    - There was a security patch released a month earlier but many customers had not installed the update
    - The payload defaced the web site then the further action depended on the day of the month
    - On day 1-19 it would look for more IIS servers
    - On day 20-27 it would launch DOS attacks on several fixed IP addresses like the white house

- As you can see vulnerabilities are often caused by buffer overflows

    - In case of the “Code Red” worm a long query parameter caused the buffer overflow
    - GET /default.ida?NNNNNNNNNNNNNNNNNNNNNNNNN
      NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
      NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
      NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
      NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
      NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
      NNNNNNNNNNNNNNNNNN
      %u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801
      %u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3
      %u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a HTTP/1.0
    - The string of N’s is used to overflow the buffer
    - After the N’s is where the payload is, which is interpreted as computer instructions that propagate the worm

- In August 2001 there was another worm called “Code Red 2”
    - It was completely different software but it exploited the same vulnerability
    - The payload was boot-resistant backdoor (more below)
    - The targets were pseudo-randomly chosen - targets in own subnet were preferred
- How did the Code Red 2 Backdoor work?
    - It replaced “explorer.exe” with a modified copy on C: and D: drives
    - It copied “cmd.exe” to special directories
    - It modified the registry to give special directories full system access
    - Full access is now possible by running cmd.exe in special locations
    - The worm continues to run in the background and resets the registry entries every 10 minutes

- The nimda (“admin” backwards) worm surfaced in September 2001
    - It was the most widespread virus/worm in just 22 minutes
- It used multi-mode propagation - in total 5 methods
    - IIS servers from infected clients
    - Direct copy via open network shares
    - Email + virus payload to address book
    - Modifying web pages on host server (attempt to infect web server clients using JavaScript exploit)
    - Code Red 2 backdoor

- Next, the SQL Slammer worm was spread in January 2003
    - It exploited a buffer overflow in Microsoft SQL server
    - There was a patch to fix this which had been released six months earlier…
    - It infected 90% of vulnerable hosts in 19 minutes
    - The worm was very small: Just 376 bytes, it fits inside a single IP packet and hence can use UDP

    - Many internet routers collapsed under very high traffic caused by the worm - over 15% packet loss at peak

- The Conficker worm is from November 2008 but is still active because the propagation strategy continually changes
    - It is estimated that over 1 million systems are still infected
    - It propagates using flaws in Windows software and dictionary attacks on admin passwords
    - Even some military systems got infected!
    - It used various self-defence strategies such as disabling Windows update, killing anti-malware software. It also scans every second to detect patches or diagnostic utilities
    - Later versions are designed to install botnet client

Bots and Botnets

- They were first detected in 2001 and are a serious problem since 2009
- Once a host machine has been compromised by malware (e.g. a virus or worm) it installs malware to become a botnet client (also called bot or zombie)
    - A very large number of such bots will be created
    - A botnet master (or bot herder) will send commands to the clients from a command-and-control (C&C) server
    - C&C servers were originally using IRC (Internet Relay Chat) but today they usually use HTTP

- A botnet is typically used for
    - DDOS attacks
    - Sending email spam
    - Running spyware
    - Hosting illegal content
    - Mining bitcoins
- A botnet can be hired out to other criminals
    - An entire black economy was created
    - There is a lot of infrastructure and market places
- Modern botnets now use P2P (peer-to-peer) messages, distributed and replicated C&C servers, greater emphasis on stealth to avoid detection - and they can detect attempts to disable the network

Adware and Spyware

- Adware is any software advertising something
    - It is legitimate if it does so with the user's consent and knowledge
    - However, it is malware when it is unwanted

- Can be just annoying but also be a more serious problem
    - What if user activity is recorded and reported back as well (spyware)?

Scareware

- “We have detected that your computer has been infected with a virus! Click here for a removal package”
    - This is how scareware can get onto your computer
- Usually the claimed virus is bogus - as is the downloaded package
- The victim either pays for the worthless package or unwittingly installs dangerous malware

Hoaxware

- “There is a dangerous virus going around! Quickly, email all your friends to warn them about his…”
    - A typical example of hoaxware
- Here is an example
    - “I got this message about a virus that can produce lot of dammage [sic] to your computer. If you follow the instructions which are very easy, you would be able to "clean" your computer. Apparently the virus spreads through the adresses book .
      I got it, then may be I passed it to you too, sorry. The name of the virus is jdbgmgr.exe and is transmitted automatically through the Messanger and addresses book of the OUTLOOK.
      The virus is neither detected by Norton nor by Mc Afee. It remains in lethargy ("sleeping") for 14 days and even more, before it destroys the whole system. It can be eliminated during this period.”

Ransomware

- Ransomware restricts access to infected computers - or their files
    - The computer (or files) is held hostage
    - Payment is required to criminals to restore access
    - Money needs to be untraceable so usually bitcoins, western union or premium-rate telephone numbers are used
- Simple ransomware is just a little more than scareware
- More advanced ransomware is a serious problem
    - Files are encrypted with “hard” cryptography
    - You better have a backup otherwise you need to pay

- An example is Cryptolocker
    - The control server creates a 2048-bit RSA key pair and sends the public half to the malware
    - The malware generates a new 256-bit random key for each file on the computer, encrypts the file using 256-bit AES, encrypts each random key with the asymmetric public key and stores the encrypted key before deleting the unencrypted random key and the original files
    - Ransom of 2 bitcoins (around $2k in 2013) was required to release the private key
- Additional examples are
    - Petya – 2016 – spread via infected email
    - Wannacry – 2017 - $300 ransom – 7 day deadline
    - 'EternalBlue' exploit (ex-NSA, bug in Windows SMB protocol – patched by Microsoft) plus backdoor infected 230,000 computers in 150 countries
    - Notpetya – 2017 – $10B damage worldwide
      multiple spreading techniques including EternalBlue
      Not ransomware – files can't be recovered
    - Bad Rabbit – 2017 – bogus Flash update

Backdoor/Trapdoor

- A backdoor is anything that avoids the normal authentication mechanism
    - Hence it allows for illegal access without a password
- An example is the Ken Thompson Unix backdoor
    - Check the section about the Trojan Horse above

Keylogging

- Keylogging or keystroke logging means recording keystrokes as keys are pressed on a keyboard - without the user being aware
    - They can capture passwords as they are typed
    - Different mechanisms can be used to log key strokes - software and hardware
- Each keylogger needs a way to communicate captured keystrokes to whoever installed the keylogger without being detected

- A similar logging mechanism called "Tempest" can detect computer activity by the electromagnetic radiation emitted by all hardware
    - It does not just work with wireless devices
    - It does not need any hardware connection - It just needs to be sufficiently close
    - and without any hardware connection
    - It is not a new technology - military and national security agencies have been actively using it since the 1970's

    - This is why military etc. computers are specifically hardened to prevent this attack
    - “Tempest” now usually refers to the entire field of emission security

Rootkits

- A rootkit is software that allows for continued privileged access to a computer
    - It’s usually malware but not always (can be installed by an administrator for other purposes)
- The key objective is stealth
    - The existence of the rootkit and payload are hidden and undetectable
- It typically modifies the OS kernel together with system tools capable of detecting changes
    - For example, in unix there would be modified versions of ps, who, passwd etc
- The removal of rootkits is difficult
    - You might as well just reinstall the OS from scratch

Additional forms of malware and topics

- Drive-by download
    - You download something but what you don’t notice is that you download something else as well - the malware
- Zero-day exploit
    - The software vendor has not detected the vulnerability, is working on a patch or released a patch but users haven’t installed it yet
- Spear phishing
    - A form of phishing that is specifically tailored to an individual using social engineering techniques
- Man-in-the-middle attacks
    - Intercept communication “in the middle” - between sender and receiver - and make sure that they don’t notice
- Side-channel attacks
    - An attack based on information gained from the implementation of a system rather than weaknesses in the implementation
- Timing attacks
    - The time a system takes to process data depends on the amount of data that is being processed
    - An attacker can thus gain information about the amount of data
- Covert channel
    - A communication channel which allows to transfer information out of a secure system
- Logic bomb (and time bomb)

    - Malware that is designed to cause damage under certain conditions, e.g. on a certain day
- Virus detection
    - Early anti-virus software used signature detection
    - However, modern viruses are polymorphic - the virus modifies itself to defeat signature detection
    - Nowadays, multiple detection techniques are used
    - Examples are sandbox detection and data mining to classify behaviour
    - No algorithm can perfectly detect all possible viruses
- In IPv6 address scanning is no longer attractive because the the large address space

Introduction to Cryptography

- Cryptography is “the art and science of encryption”
- This section will be non-mathematical but you still need a basic understanding of
    - modular arithmetic
    - number systems
    - exponentiation
    - matrix multiplication - remember that in general M1 * M2 is not equal to M2 * M1
- Cryptography alone is not the solution....
    - It is often mis-used
    - People use strong cryptography but all that the bad guys have to do is use the back door
- “A security system is only as strong as its weakest link”
    - Yes, this is the 3rd time I mention this
    - You often find the digital equivalent of a bank vault fitted to a tent
    - Designers of secure systems need professional paranoia - need to design against the bad guys

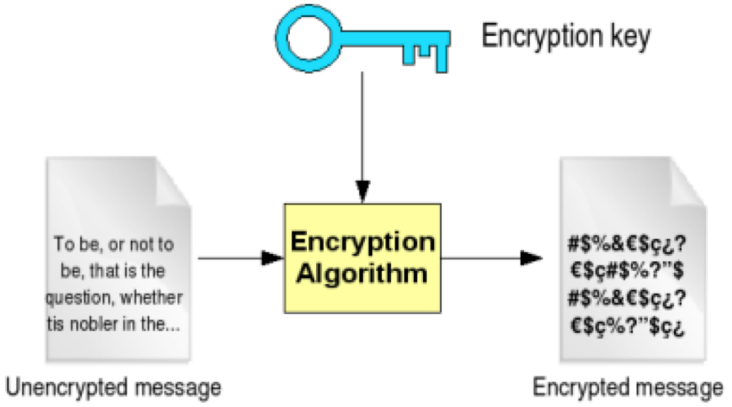
- The primary security issues are
    - Confidentiality: The message has to remain private
    - Integrity: The message must not be modified
    - Authentication: Confidence in the identity of the parties involved
    - Non-repudiation: Parties cannot deny having generated the message/data - often includes a timestamp - this is evidence acceptable to a court of law

- Some terminology that will be used in this section...
    - P or P/T: plaintext
    - C or C/T: ciphertext
    - K: key
    - P → C: process of encryption (enciphering)
    - C = E(K, P): getting the ciphertext from encryption using a key
    - C → P: process of decryption (deciphering)

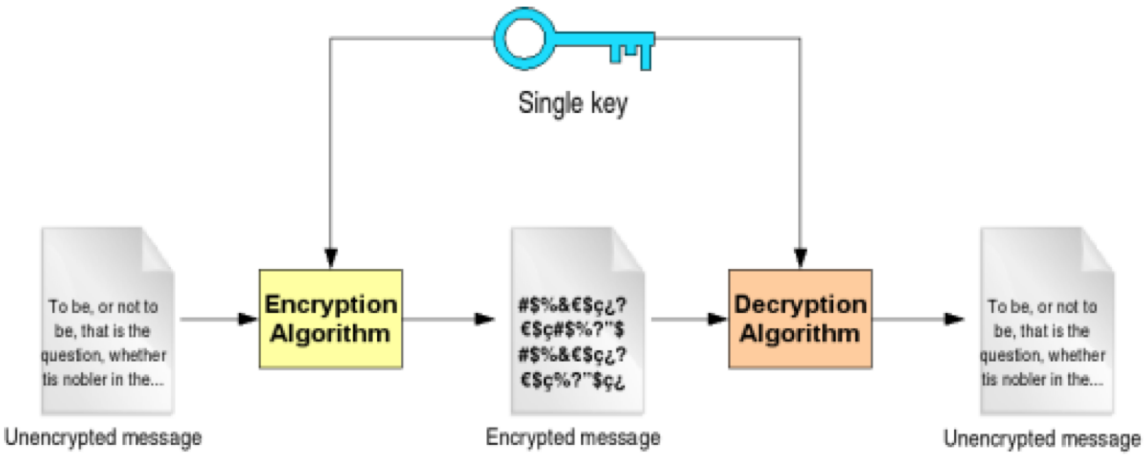
- P = D(K, C): getting the plaintext from decryption using a key
- For symmetric encryption, P* = D(K, E(K, P)) and P* = P
  You'll get the same plaintext back if you first encrypt it and then decrypt the result

Encryption - symmetric and asymmetric

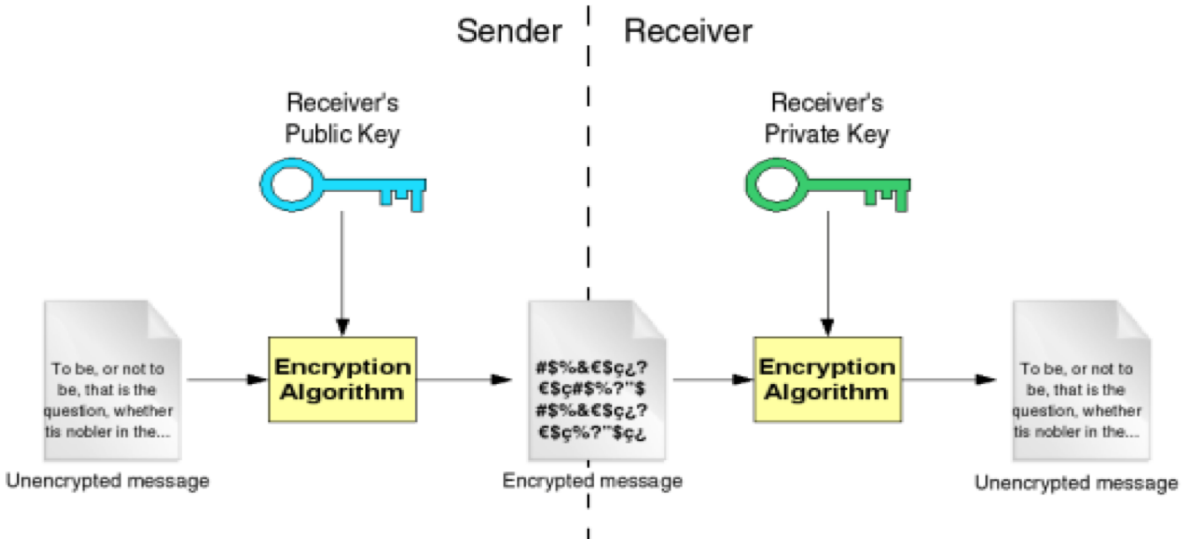
- Encryption
  - Is done for confidentiality and integrity
  - Using an encryption algorithm and an encryption key you go from plaintext (unencrypted message) to ciphertext (encrypted message)
  - For a given plaintext message, the encrypted version will differ for different keys (if you use the same algorithm)



- Symmetric encryption
  - Is also known as "secret key"
  - It uses the same key for encryption and decryption
  - There's only one key known to the sender and recipient
  - It's very easy to implement and very fast but also quite vulnerable because how do you safely exchange the key?
  - Examples are DES, 3DES, AES



- Asymmetric encryption
  - Is also known as “public key”
  - It uses a private and a public key - that’s 2 keys compared to only 1 key in symmetric encryption
  - Messages are encrypted using the public key and decrypted with the private key (or vice versa for the digital signature - more about it later)
  - The private key is locked away but the public key is distributed (it’s not secret)
  - It’s significantly slower than symmetric but more secure
  - An example is RSA



- A main problem is the key distribution problem
  - It’s clearly a problem with symmetric keys: Sender and recipient need the same key but how do they safely exchange it?
  - It’s also a problem with asymmetric keys: Here the question is how do you know that a key you receive is from the person you are expecting it from?
  - How do you know who you can trust?
- Often, a symmetric session key is used
  - It is randomly generated (not pseudo-random, more about that later)
  - It’s only used for a short time
  - The symmetric key is exchanged using a public key system

Historical and simple ciphers

- The earliest cypher is from Julius Caesar
  - It’s an example of a monoalphabetic substitution cypher: Take letters and replace them with other letters
  - In this case, the alphabet is shifted by a certain number the sender and recipient agrees on
  - For example, if we agree on 3 and the P/T is “BAD”, then the C/T is “EDG”

- It is highly susceptible to cryptanalysis
    - If you know the language used you know which letters are most common, thus you can guess what the shift is
    - If you don't have a clue you can just generate 26 plain texts and find the one that makes sense (brute force approach)

- Slightly more complex is the affine cypher
    - It's still a monoalphabetic substitution cypher
    - The key is made up from two numbers (a, b)
    - Each character in the P/T is mapped to a number, e.g. A..Z is mapped to 0..25
    - The variable m denotes the number of characters in the alphabet, in this case m = 26
    - For each character P_X we are going to create an encrypted character C_X as follows: $C_X = (aP_X + b) \, mod \, m$
    - a and m must be co-prime, which means that the only positive integer that divides both must be 1
    - If you choose a = 1 you get the Ceaser cipher
    - There are just 286 non-trivial affine ciphers for the English alphabet

- How many possible alphabets are there available for monoalphabetic substitution ciphers?
    - If the letters can be in any position, then there are 26! possible alphabets available for use
    - Surely such a large number would make cryptanalysis (cracking it) very difficult?
    - No, because a cryptanalyst has many techniques available to reduce the number of possible alphabets drastically
    - For example, as mentioned above, if you know the plaintext is in English you also know that certain letters are much more frequent than others
    - In English, E, T, O and A are more frequent than J, Q, X, Z

Codes and ciphers

- What is the difference?
    - For a cipher it is always true that the lengths of the plaintext is the same as the length of the ciphertext
    - Codes work differently, they require a dictionary or look up table instead of a key
    - If the codes are shorter than plaintext then you can use it for compression
    - An example is the Huffman coding which is used for compression, not security

- How can you break codes?

    - Just steal the codebook - but don't take the original, make a copy!
    - Intercept messages and try to find out what the codes are

One-Time Pad (OTP)

- That's the only unbreakable cipher
    - It's also called "Vernam cipher"
    - It was first proposed in 1882

- It uses exclusive or (XOR) with the key to go from P to C and from C to P
    - Here is how XOR works

input1	input2	output
0	0	0
0	1	1
1	0	1
1	1	0

    - The cool thing about XOR is that you can take any of the two columns above, XOR them and you'll get the remaining column!
- Therefore, if XOR is used twice you get back the original text
    - P XOR K = C
    - C XOR K = P
    - P XOR K XOR K = P

- There are three requirements
    - The length of the key must be equal to (or greater than) the length of the plain text
    - The key must be random, not pseudo-random (more about that later)
    - The key must never be reused

- The key length is a major disadvantage
    - It must be as long as the plaintext
    - How do you securely distribute the key? (key distribution problem)
    - If you start reusing the key it's no longer unbreakable

Random Numbers

- All computers (or programming languages) have a random number generator (RNG) or pseudorandom number generator (PRNG)

- A PRNG cannot be used for any cryptographic application, because it is deterministic

    - It generates random numbers based on a seed you give it once at the beginning
    - Deterministic means that it always repeats the same sequence of random numbers
    - Thus you can reverse-engineer to discover the algorithm being used

- We require a truly-random process, e.g. thermal noise in electronic device, radioactive decay, roulette wheel etc.
    - The first UK lottery in 1957 used thermal noise
    - More recently, machines for the UK national lottery select random numbered balls
- There also exists a cryptographically secure pseudo-random number generator (CSPRNG), which we can use
    - It starts with a random input from a high-quality source and essentially expands it
    - One must not be able to reverse-engineer

Data Encryption Standard (DES)

- What's the difference between a stream and a block cipher?
    - Stream cipher converts one symbol at a time
    - Block cipher converts a group of symbols together

- How to achieve a secure cipher?
    - Have confusion and diffusion!
    - Confusion means that there is a complex relationship between the C/T and the key - there is no obvious connection
    - Diffusion means that the P/T structure is spread over the whole C/T - change one bit in the P/T and on average half the C/T bits change
    - Both is usually achieved by a series of repeated substitutions and permutations

- The Data Encryption Standard (DES) was developed by IBM in 1977
    - It is a symmetric cipher
    - It is a block cypher with 64-bit blocks and a 56-bit key
    - It had a major influence on modern cryptography

- DES was originally designed for 10 years but it was only replaced in 2002 by AES (Advanced Encryption Standard)
    - US Government agencies were required to switch from DES to AES
    - DES is still perfectly acceptable for many legacy applications
    - In most of those cases triple DES is used

- The algorithm was published but that does not make it insecure!
    - Why? Kerckhoff's principle

    - Kerckhoff's principle says that "A cryptosystem must be secure even if everything about the system (apart from the key) is public knowledge"
    - The entire cryptographic community worldwide has been trying to break DES for over 40 years with very little success
- If the algorithm was secret it would not make it more secure
    - There are many examples of secret systems that were broken despite the secrecy
    - An example is the Japanese Purple cipher in WW II

- Nowadays DES is not considered secure because of the short key length of 56 bits
    - Despite the short key length there is still no real algorithmic weakness known after 40 years of study...
    - There exists Triple DES (3DES/TDES) which remains effectively unbreakable (hence secure)
    - AES has a minimum key size of 128 bits to address this issue

- How can you attack (the normal) DES?
    - One possible attack is to brute force it
    - Assume that we know both P/T and C/T
    - The key can always be discovered through trying out all possible keys - search the entire key space
    - DES has 2^56 possible keys
    - A DES cracker that searches the entire key space was proposed in 1977 and built in 1998
- Nowadays, the focus of security attacks shifts to obtaining keys

- How long does it take to test the DES key space?
    - The key is 56 bits long so we require up to 2^56 tests
    - On average we will find the key after testing half the keys
    - Assume 1 million tests every second
    - There are 31*10^6 seconds in a year, hence an exhaustive search would take 2000 years
    - However, the DES cracker runs 43,000 of those tests in parallel
    - A key length of 56 bits was long enough in 1977, but not with modern hardware…

- Another way to attack many ciphers is called differential cryptanalysis
    - It was invented by Biham and Shami in 1990
    - This method attempts to discover non-random behaviour by changing the input and exploiting this to recover the key
    - It was successful against many ciphers in common use
    - However, DES cannot be cracked this way
    - DES was designed to protect against such attacks
    - Both IBM and NSA knew about the technique in 1974 but kept it secret

- Can we make DES more secure by using double DES?

    - We just encrypt twice - encrypt with 1 key, then encrypt again with another key
    - Do we get an effective key length of 56 * 2 = 112 bits?
    - No! The effective key length is only 57 bits because of the “meet in the middle” attack (see below)

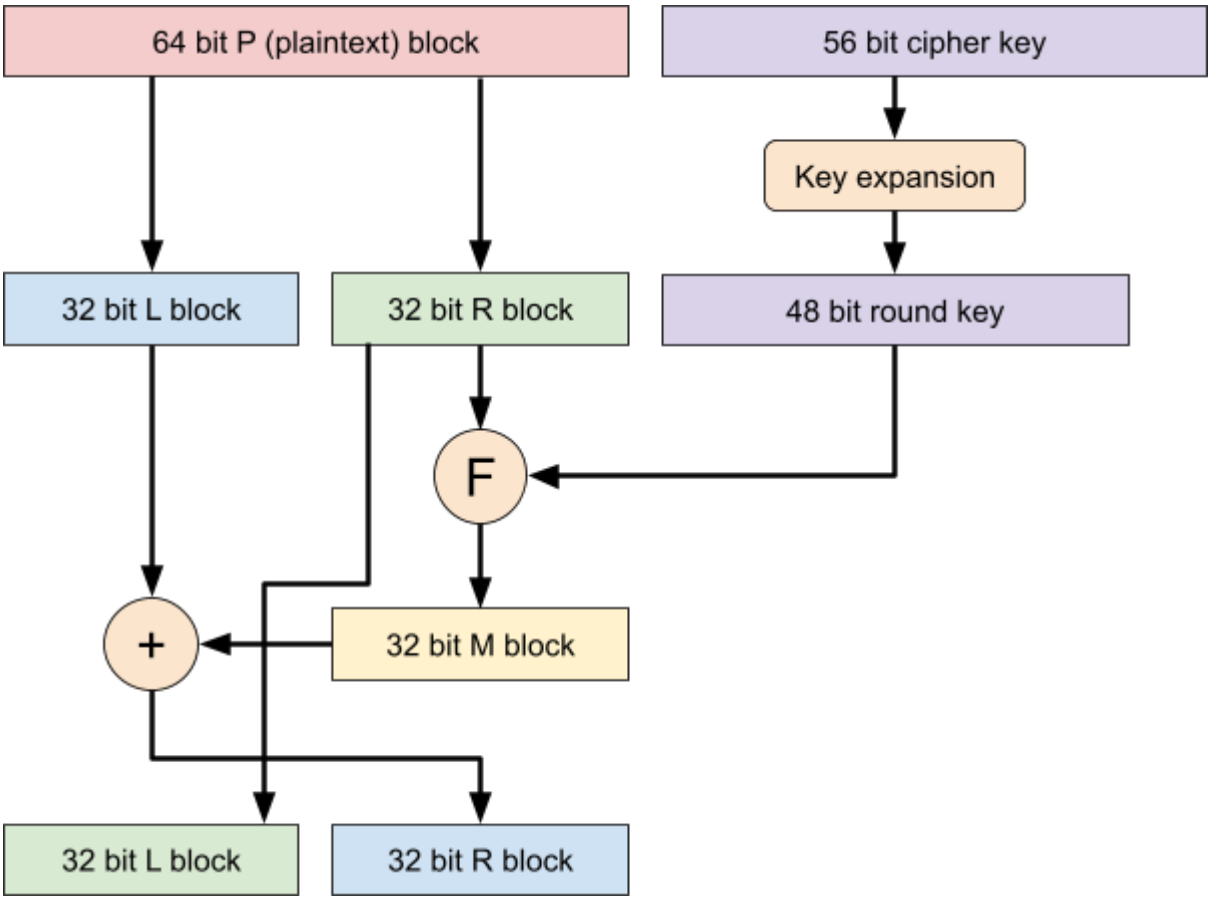
- How does the meet in the middle attack work?
    - We assume that we know both the P/T and the C/T (“known plaintext attack”)
    - Take the P/T and encrypt it by using all possible keys (2^56), store all resulting intermediate ciphertexts
    - Take the C/T and decrypt it by using all possible keys (again 2^56) and look for a match with the intermediate ciphertexts that are stored
    - Therefore, we need to run up to 2 * 2^56 tests, which is the same as 2^57
    - And yes, we do need a huge storage space for the intermediate texts

- So double DES is not secure, what can we do to make DES more secure?
- We use triple DES!
    - We can either use 3 different 56-bit keys to obtain an effective key size of 112 bits
    - Or we can use 2 different 56-bit keys to obtain an effective key size of about 80 bits
- Triple DES is secure and currently widely used for financial transactions
    - It is estimated that it is secure up to about 2029
    - For many government agencies it will be disallowed after 2023

- How does DES actually work?
    - It uses a combination of permutation and substitution
    - The permutation uses a so called P box with 32 inputs and 32 outputs
    - The substitution uses a S box with a 6-bit input and a 4-bit output - we will use 8 S boxes in parallel

- The algorithm used is a Feistel cipher
    - Used in many block ciphers - but not AES
    - The following bit lengths are specific to the DES implementation
    - First, we split the plaintext into 64 bit P (plaintext) blocks
    - We take each of the plaintext blocks and split it into a 32 bit L (left) and 32 bit R (right) block
    - Now, we are going to start the first round (iteration)
    - We take the 56 bit key and generate a 48 bit round key through key expansion
    - We feed the R block and the round key into a round function F to generate a 32 bit M block
    - We XOR the L and M block and get a new R block for the next round
    - We just copy over the old R block and it becomes the new L block for the next round

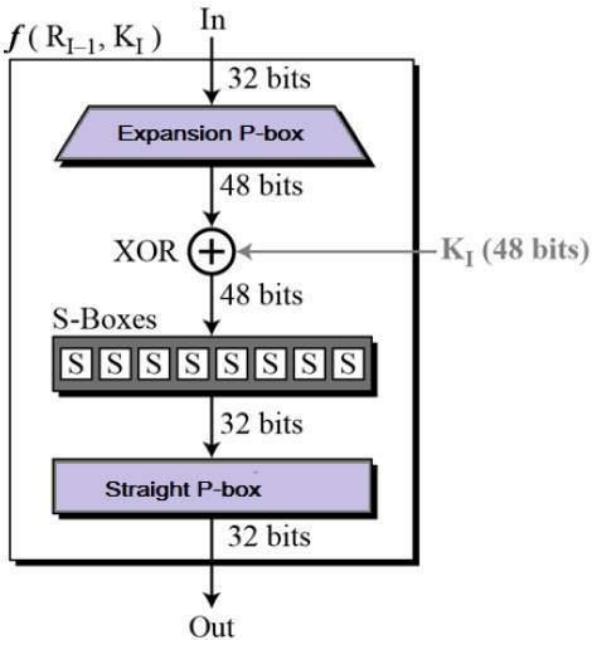
    - Now we are done with the first round, and we repeat this process 16 times since there is a total of 16 rounds



- How can we decrypt?
    - Look at the graphic above, you can see that the bottom L block and the top R block (both in green) are the same!
    - Using the cipher key we can generate the round key
    - With the round key and the top R block we can get the M block
    - Now we XOR the M block with the bottom R block to get the top L block
    - We repeat this process 16 rounds to get the plaintext

- How do we get the round keys?
    - The 56 bit cipher key is expanded into 16 48 bit round keys
    - There is one round key for each round

- How do we get the M block (i.e. what is the round function F)?
    - The 32 bit R block is expanded into 48 bits and XORed with the round key
    - The 48 bit result is split into 8 sets of 6 bits, these are the inputs to the S boxes
    - Each S box has a 4 bit output so we end up with 32 bits from all S boxes
    - These 32 bits go through the P box
    - The output of the P box is the M block



- S and P boxes are unchanged between the rounds
    - The S box provides a “one way” function
    - The 4 bit output of the S box could have come from any of four different 6 bit inputs
    - The S box function is similar to hashing (N inputs map to 1 output)

Advanced Encryption Standard (AES)

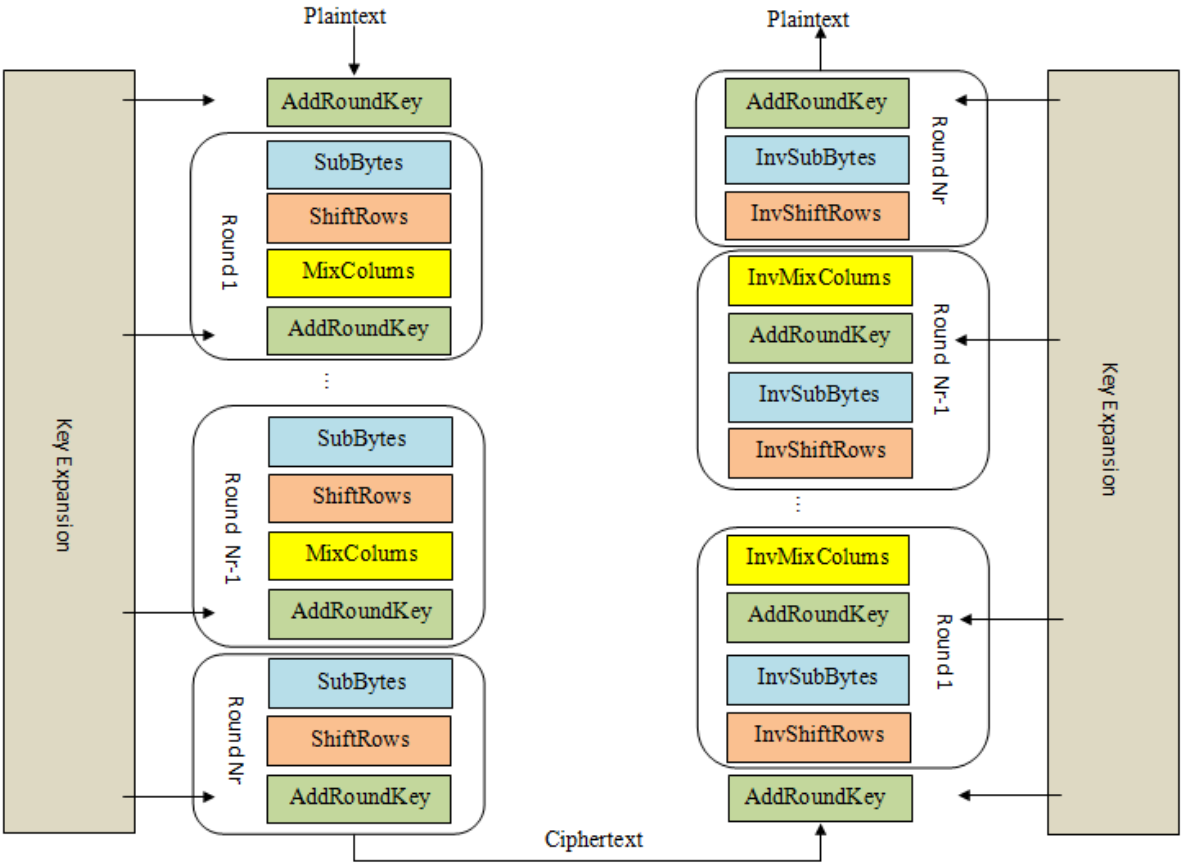
- In 1997 there was an open call for proposals for a DES replacement
    - There were five finalists
    - The winner of the competition was Rijndael
    - The main reason was that it was faster than the other competing algorithms

- AES is a symmetric block cipher
    - The blocks are 128 bits each
    - The key can be either 128, 192 or 256 bits
    - It uses substitution and permutation like DES does, however, the S and P boxes work completely differently

- Each 128 bit block is also known as the “state”
    - It is represented as a 4x4 grid (or matrix)
    - Each cell in the grid is one byte

- AES encrypts/decrypts text in a number of rounds
    - There are 10/12/14 rounds, depending on the key length
    - We generate a bunch of round keys from the original key
    - Before the first round, we XOR the initial round key with the plaintext

- For each round we substitute bytes (S box), shift rows, mix columns (both P box) and XOR the current round key
- The last round is an exception - We do not mix columns
- Each step is reversible so we can decrypt by running the algorithm backwards



- How does substituting the bytes work (SubBytes)?
  - The S box provides confusion through substitution
  - Each byte (entry in the grid) is looked up in a table
  - There are 256 entries in total

- How does shifting the rows work (ShiftRows)?
  - We take the state and shift each row by a certain amount of cells to the left
  - When we shift the leftmost cell to the left, it becomes the rightmost cell
  - How far do we shift each row?
  - We shift row 0 (top row) by 0 cells (stays the same)
  - We shift row 1 by 1 cell, row 2 by 2 cells and row 3 by 3 cells

- How does mixing the columns work (MixColumns)?
  - Each column of the state is a vector
  - We multiply a matrix M with each column vector to get a new column
  - [new_column] = [M] [old_column]
  - For decryption we use the inverse matrix M'
  - We use Galois Field arithmetic - GF(2^8) or GF(256), more details below

- How does adding the round key work (AddRoundKey)?
    - There is a round key for each round, plus an initial round key that is applied before the first round
    - We generate these round keys from the full length key, which is called key expansion
    - We add the round key to the current state but perform this addition in a Galois Field (GF, more below)
    - If we are in GF(2^k) - more details below - the addition operation becomes an XOR operation

Galois Field (GF)

- We consider operations on a set with a finite number of elements as operations on a finite field or Galois field (GF)
    - It is defined as GF(p^k) where p is a prime number and k is a positive integer
    - A GF contains numbers from 0 to (p^k) - 1

- A field is a set in which addition, subtraction, multiplication and division are defined
    - When we do operations on a GF we do them as in regular arithmetic but we may have to reduce the result if it lays outside the GF
    - The reduction function we apply depends on the value of k
    - If k = 1 then we use modulo p as the reduction function (see example below)
    - For cases other than k = 1 we need to reduce through the subtraction of a prime (or irreducible) polynomial - more about it later

- For example, take the set {0, 1}, which is called GF(2^1) or just GF(2)
    - We will use mod 2 as the reduction function because k = 1 and p = 2
- GF(2) addition is the XOR function
    - Take two numbers a and b from the GF and calculate a + b
    - If the result is >= 2 then take it mod 2
    - For instance, 1 + 1 = 2 but 2 is not in our GF so we need to reduce it using mod 2
    - 2 mod 2 is 0, which is our final result
    - Therefore, in GF(2), 1 + 1 = 0
    - You'll see that the result is the same as if you do a XOR b
- GF(2) subtraction is XOR as well
- GF(2) multiplication is AND
- GF(2) division is AND as well except that it is undefined for a zero divisor

- In the OTP (one-time pad), the encryption is the GF(2) addition of the key whereas the decryption is the GF(2) subtraction of the key

- How about GF(2^k) and k is not 1?

- This is slightly more complicated compared to when k = 1 (see above)
- Because p is 2, all addition and subtraction is done using the same operation as in GF(2), which is XOR
- However, the result is still in GF(2^k) which means it can be any number between 0 and 2^k - 1
- Multiplication must be by “shift-and-add” (explanation follows in picoAES8)
- If we need to reduce then we need to subtract a prime (or irreducible) polynomial, which means subtracting some binary

picoAES5

- This is an ultra-simplified version of AES
  - The state is just 2x2 and each element is a number between 0 and 4
  - All arithmetic uses GF(5)
  - Shifting the rows is just swapping the elements in the bottom row
  - For example, for mixing the columns we will use the following matrices

$$[M]: \begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix} \qquad [M']: \begin{bmatrix} 2 & 2 \\ 4 & 2 \end{bmatrix}$$

- **Note that picoAES (this and the other versions below) is not an official standard**
  - It is made up to demonstrate the core operations of AES
  - If you see it in an exam it might be defined in another way
  - Thus, I don’t think there is any point in memorising matrices or S boxes

picoAES7

- Another ultra-simplified version of AES
  - This is the same as picoAES5 but in GF(7)
  - Note that GF(7) is GF(7^1), so p = 7 and k = 1, which means that the reduction function is modulo 7

- One round of encryption is defined as follows
  1. SubBytes
  2. ShiftRows
  3. MixColumns
  4. AddRoundKey
- Apply the inverse operations backwards for decryption

- For the following example, we are given
  - The following S box - for instance, 1 will be replaced by 4

2	4	3	5	6	1	0

- The mix column matrix and its inverse

M =

2	3
1	2

M' =

2	4
6	2

- The round key

1	2
3	4

- For the given plaintext state below, what is the state after one encryption round?

1	2
3	4

1. **SubBytes**
   We substitute the numbers of the state using the S box provided above to get a new state
   To substitute a number "n" take the nth index of the S box

1	2
3	4

substitute

2	4	3	5	6	1	0

=

4	3
5	6

2. **ShiftRows**
   We shift the bottom row by one to the left (swap elements) to get a new state

4	3
5	6

shift =

4	3
6	5

3. **MixColumns**
   We multiply the mix columns matrix M with each column of the state to get a new state
   If any result is >= 7 we need to mod 7 it
   First, we start with the left column of the state

2	3	*	4	=	26	mod 7 =	5
1	2		6		16		2

We continue with the right column of the state

2	3	*	3	=	21	mod 7 =	0
1	2		5		13		6

We combine both columns to get the new state

5	0
2	6

4. **AddRoundKey**
   We add the round key to the state
   If any result is >= 7 we need to mod 7 it

5	0	+	1	2	=	6	2	mod 7 =	6	2
2	6		3	4		5	10		5	3

final state

- For the given ciphertext state below, what is the state after one decryption round?

5	3
6	2

1. **InvAddRoundKey**
   The inverse of AddRoundKey, so we need to subtract the key from the state
   If any number becomes < 0 we need to add 7

5	3	-	1	2	=	4	1	→	4	1
6	2		3	4		3	-2		3	5

2. **InvMixColumns**
   Same as MixColumns except that we use M' (the inverse of M we used in MixColumns) for the multiplication
   First, we start with the left column of the state

2	4	*	4	=	20	mod 7 =	6
6	2		3		30		2

   We continue with the right column of the state

2	4	*	1	=	22	mod 7 =	1
6	2		5		16		2

   We combine both columns to get the new state

6	1
2	2

3. **InvShiftRows**
   Same as ShiftRows except that we shift in the opposite direction (right not left)
   For picoAES it doesn't matter whether we shift left or right as we just swap the bottom two elements

6	1	shift =	6	1
2	2		2	2

4. **InvSubBytes**
   Same as SubBytes except that we substitute in the other direction
   Therefore, we substitute the number through the index of where it appears in the S box

6	1	inverse substitute							=	4	5
2	2	2	4	3	5	6	1	0		0	0

final state

picoAES8

- A simplified version of AES
  - The state is just 2x2 and each element is a number between 0 and 7
  - All arithmetic uses GF(2^3) or GF(8)
  - All addition and subtraction is done in GF(2)
  - For multiplication, we need to use an irreducible polynomial for reduction - don't worry if you don't understand what this is, I'll explain the process below

- In order to do multiplication, we are going to create a lookup table for multiplying two numbers x and y
  - Later, when we are encoding/decoding and need to multiply two numbers, we use the table to look up the result of the multiplication
- Without further ado, let's compute the lookup table for two numbers x and y
  - We compute the first two rows (row 0 and row 1) as usual
  - These two rows are essentially our base cases for building up the table
  - They don't require any reduction because we never exceed 7

x/y	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7

  - From now on we will build upon the base cases and rows we have already computed

  - If we are in row n and n is an even number then we take the results from row n/2 and multiply those by 2
  - Multiplication by 2 is done by shifting one bit left
  - So how do we compute row 2?
  - We take the results from row 1 (2/2) and multiply those by 2 (shift 1 bit left)
  - In the explanation below, "←" means shift one bit left
  - If the result after shifting is >= 8 we need to reduce by subtracting (XORing) the irreducible polynomial x^3 + x + 1, which is 1011 in binary
  - The reduction is indicated by the "-" sign in the table below

x/y	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
2	0	2	4	6	3	1	7	5
Expl.		0001 ← 0010	0010 ← 0100	0011 ← 0110	0100 ← 1000 - 1011 =	0101 ← 1010 - 1011 =	0110 ← 1100 - 1011 =	0111 ← 1110 - 1011 =

					0011	0001	0111	0101

- If we are in row m and m is an odd number then we take the results from row 1 and add the results from row m-1
- Therefore, to compute row 3 we are going to add results from row 1 and row 2 (3-1)
- Adding is done using a bitwise XOR operation, which basically comes down to adding two numbers but dropping the carry bit if there is any

x/y	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
3	0	3	6	5	7	4	1	2
Expl.		0001 + 0010 = 0011	0010 + 0100 = 0110	0011 + 0110 = 0101	0100 + 0011 = 0111	0101 + 0001 = 0100	0110 + 0111 = 0001	0111 + 0101 = 0010

- We can continue to follow this procedure until we computed the whole table

y:\x=	0	1	2	3	4	5	6	7	notes:
0	0	0	0	0	0	0	0	0	R0: trivial
1	0	1	2	3	4	5	6	7	R1: trivial
2	0	2	4	6	3	1	7	5	R2: shift R1, reduce?
3	0	3	6	5	7	4	1	2	R3: R1+R2
4	0	4	3	7	6	2	5	1	R4: shift R2, reduce?
5	0	5	1	4	2	7	3	6	R5: R1+R4
6	0	6	7	1	5	3	2	4	R6: shift R3, reduce?
7	0	7	5	2	1	6	4	3	R7: R1+R6

- For example, we use the following S box (for substitution): [1,2,5,0,3,7,4,6]
- For shifting rows we just swap elements in the bottom row
- For example, for mixing columns we use the following matrices

$$[M]: \begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix} \qquad [M']: \begin{bmatrix} 3 & 7 \\ 4 & 3 \end{bmatrix}$$

AES continued

- In AES we use GF(2^8) or GF(256)
- Addition and subtraction is still in the underlying field of GF(2)
- For multiplication we use shift-and-add just as in GF(8)
    - As before, if the shift result ever exceeds 255 we need to reduce by subtracting a prime polynomial
    - AES uses x8+x4+x3+x+1 (0x11b)

- The following matrices are used for mixing the columns
    - M is used for encryption whereas M' is used for decryption

$$[M]: \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \quad [M']: \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 8 & 14 \end{bmatrix}$$

- "Confusion" comes from the s-box and round key addition steps, while 'diffusion' comes from the row shift and column mix steps
- Without the row shift, the column mix would always be operating on the same 32 bits – this would introduce a serious weakness
- GF(256) multiplication usually done by using look-up tables

Multi-Block Ciphertexts & Block Cipher Modes

- Most ciphertexts will require a number of blocks (e.g. 128 bit AES block = 16 ASCII characters)
- Therefore we need to split the P/T into blocks first
- To reduce the number of blocks use compression before you start encrypting
    - This practice also helps to hide the actual message length

- There are different block cipher modes to split up our P/T

- Electronic Code Book (ECB) mode
    - Do NOT use this!
    - It splits the P/T into - for example - 128 bit blocks, each gets enciphered independently
    - It cannot detect missing/duplicated/reordered C/T blocks

- Cipher Block Chaining (CBC) mode

    - Each P/T block is XORed with the previous C/T block before encrypting
    - The reverse is done when decrypting
    - What do we do about the first P/T block (we don't have a previous C/T block)?
    - We precede the C/T with an initialisation vector and use this as the previous C/T block for the first encryption
    - We want the initialisation vector to be random so that for identical P/T and identical key we get a different C/T every time we encrypt

- Output Feedback (OFB) mode
    - Generate a key stream (KS) which can be XOR'd with the P/T (similar to how the one-time pad works)
    - The key stream starts with a random initialisation vector
    - To generate the next block, we encrypt the previous block using the key
    - Repeat this procedure to generate the whole KS, which can be pre-computed
    - P/T can be sent in smaller blocks (even bytes)

- Message Authentication Code (MAC)
    - This is not about encryption but rather about the authenticity of the message
    - It's about the identity of the sender and that the message has not been modified
    - The simplest system is to encrypt using CBC, then throw away everything except the final block, which is the MAC
    - The recipient repeats calculation on the message: If the MAC computed is identical to the one sent, then the message is confirmed and also the identity of the sender (assuming that the key is secret)

Asymmetric Encryption

- With symmetric encryption, if n parties need to communicate each pair of users needs their own separate key
    - A total of n*(n-1)/2 keys are required
    - This results in a significant key management/distribution problem
    - One potential solution is using a key distribution center, another one is asymmetric encryption

- With asymmetric encryption, each user has two keys, a public and a private one
    - The public key widely distributed while the private key is always kept secret
    - We can encrypt with the public and decrypt with the private key, which is what we use to make sure that only the recipient can read the message
    - Somebody sends you a message encrypted with your public key - only you have the private key to decrypt it
    - However, we can also encrypt with the private and decrypt with the public key to ensure the authenticity of the message

    - You encrypt a message with your private key and send it and the recipient decrypts with your public key - only you could have sent it

RSA (Rivest-Shamir-Adelman)

- RSA uses asymmetric encryption
    - It's based on the difficulty of factoring very large numbers (hundreds of digits)
- It's typically 10,000 times slower than symmetric enciphering
    - This is why we can use asymmetric encryption to exchange a key which we then use for symmetric encryption
    - Key lengths for equivalent security of symmetric/asymmetric encryption cannot be directly compared - asymmetric key needs to be about ten times longer
- It was introduced in 1978
    - No flaws are known yet

- How are the public and private key created?
    - They are always created together (key pair)
    - The private key consists of two numbers d, n
    - The public key consists of two numbers e, n
    - First, choose p and q - both large primes - larger than 10^100
    - n = p * q
    - z = (p-1) * (q-1)
    - e = any number relatively prime to z (means that it has no common factor with z apart from 1)
    - d = a number such that (e*d) mod z = 1

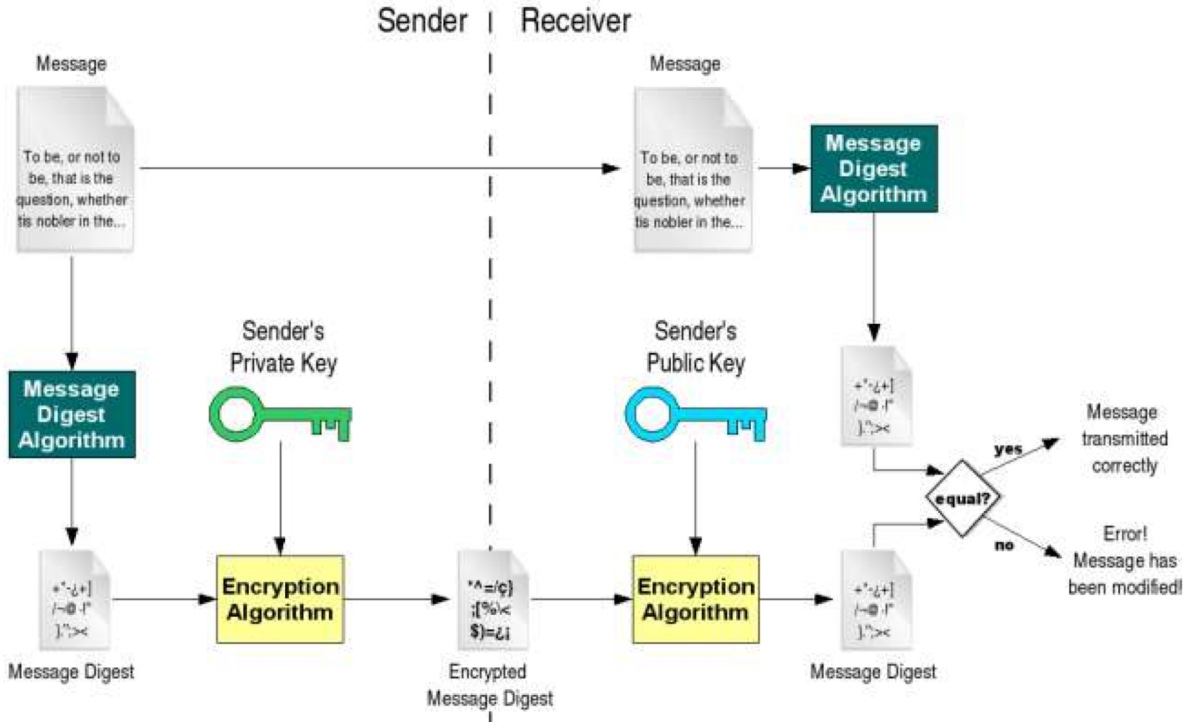
- How does encryption/decryption work?
    - Encrypt: C = P^e mod n
    - Decrypt: P = C^d mod n

Message Authentication - Cryptographic Hash Functions

- Cryptographic hash functions are also known as message digests
    - It generates a small digest or fingerprint
    - Any input change should produce a different digest
    - It's a one way function → extremely difficult to go backwards
    - It must provide good collision resistance - i.e. must be extremely difficult to find another message that produces the same digest, even though there are an infinite number of messages that all produce the same digest

- Cryptographic hash functions are the basis of the digital signature
    - They are used to ensure non-repudiation

    - e.g. electronic contract that has the same legal validity as a signed written contract (can be enforced by the courts)
- Encrypt the message digest with your asymmetric private key and publish with the document
    - Anybody can check the validity of the document by decrypting using your (published) public key and checking against the digest produced from the document itself
    - If they are identical, then the document is authentic



- Examples: MD5, SHA-1, SHA-2 (“Secure Hash Algorithm”)

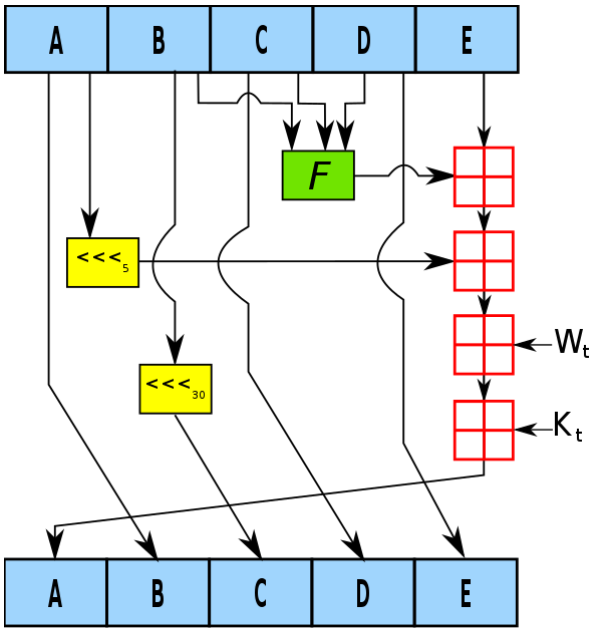
SHA-1

- SHA-1 produces a 160 bit digest
    - It was developed by the NSA and published by NIST
    - It’s very hard to find another plaintext that produces the same 160 bit hash
    - Collision attacks should require 2^80 hashes, but 2005 research suggested that only 2^69 hashes are required

- How do we compute the digest?
    - We start with 160 bit of pre-set state
    - We will divide our state into H0, H1, H2, H3 and H4 - each of these is 32 bits long
    - We consume 512 bits of our message at a time and keep changing our internal state using a compression function
    - Once we consumed the whole message we output the final state

- How does the compression function work?
  - Our first input is H0 … H4 (from the state) which we copy into A … E - we use these as a working storage
  - Our second input is a 80-word long array, which we create by expanding our 512 bit message (which is 16 32-bit words)
  - Using both inputs, we then perform 80 rounds of compression
  - Finally, we will update H0 … H4 (our state) by adding A … E to it (modulo 2^32 addition)

- One round of compression works as follows



  - <<< is a left bit rotation by the number of bits shown
  - F is a nonlinear function that varies
  - Wt is the expanded message word of round t (take the word at index t from the 80 words long array)
  - Kt is the round constant of round t
  - The red grid denotes addition modulo 2^32

- What do we do if our message length is not a multiple of 512 bits?
  - We need to add some padding to make the length a multiple of 512!
  - For the padding we need to know the length of the original message
  - We can then add padding at the end of the message as follows
  - We first add a single 1
  - At the end of the block we add the message length (in binary)
  - We fill up the empty space between with 0's

- Here are two examples that show that even a slight change in the plaintext results in a big change of the hash
- Input: "The quick brown fox jumps over the lazy dog"
  - SHA1: 2fd4e1c67a2d28fced849ee1bb76e7391b93eb12

- Input "The quick brown fox jumps over the lazy cog"
  - SHA1: de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b3
  - Even though we just changed “d” to “c” 81 bits have changed!

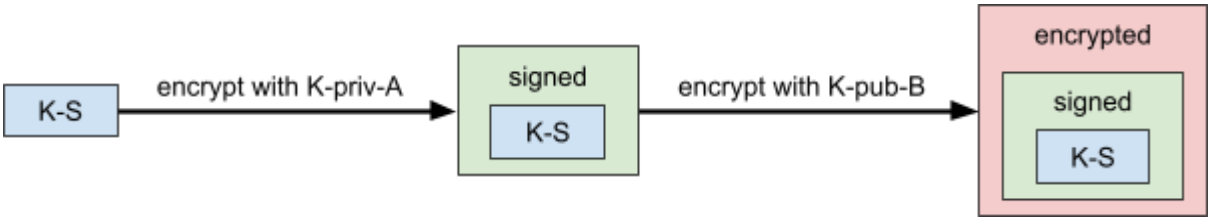
- The first SHA-1 collision was published in February 2017
  - It took around 2^60 calculations to find it - over 6,500 hours of CPU time
  - Two different PDF files that produce the same SHA-1 digest

SHA-2

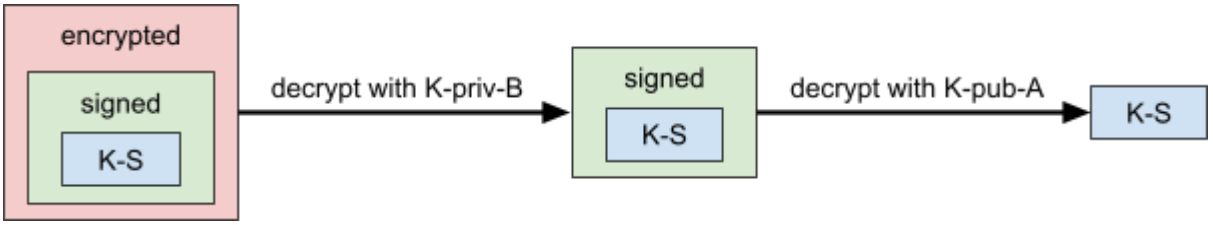
- SHA-2 is the successor of SHA-1
  - It was recommended for US government use after 2010
  - It was not widely used until early 2017 - after the SHA-1 collision was published
  - SHA-3 was defined in 2012 - just in case SHA-2 becomes unsafe to use
- It produces a digest of either 224, 256, 384 or 512 bits

Key Exchange

- Suppose you are A and want to send a symmetric key K-S to B
  - A and B have an asymmetric key pair
    A has asymmetric keys K-pub-A and K-priv-A
    B has asymmetric keys K-pub-B and K-priv-B
  - First, A encrypts K-S by using its private key K-priv-A - I will call this “signed”
  - Then, A encrypts the result by using B’s public key K-pub-B - I will call this “encrypted”



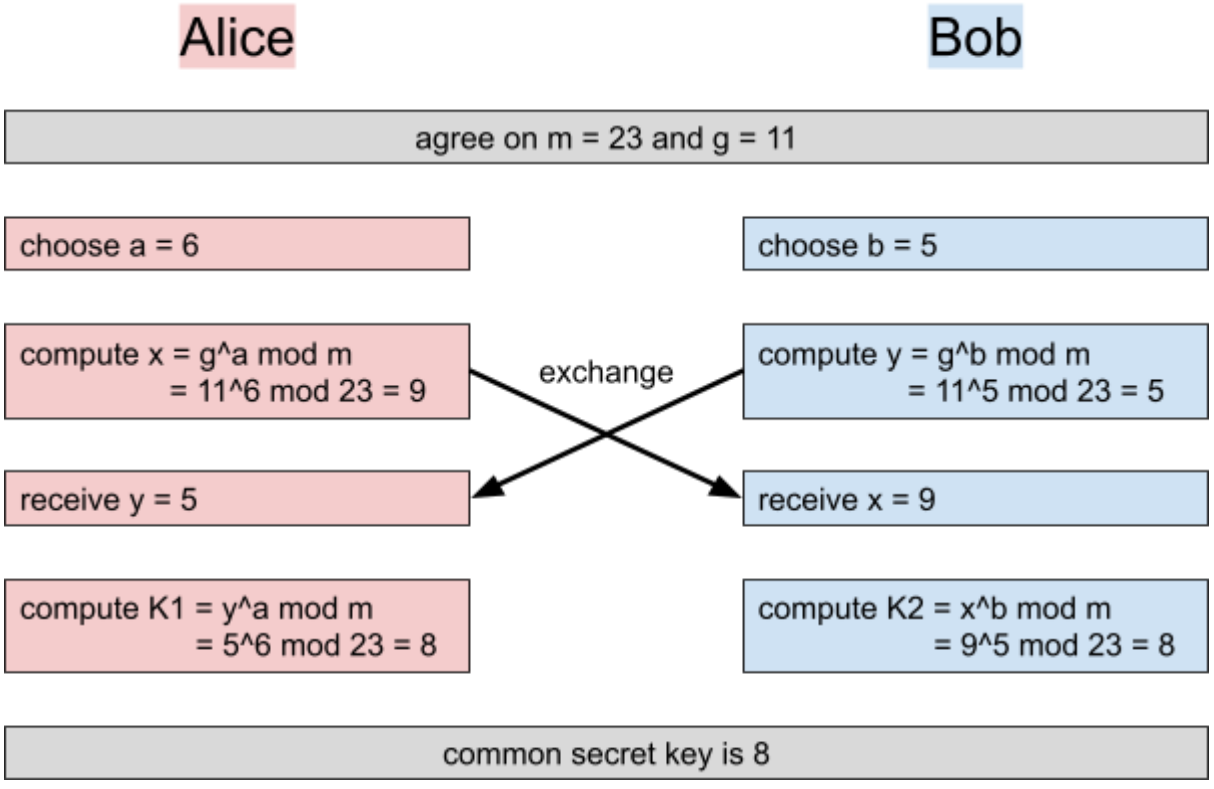
  - A sends this encrypted & signed message over to B
  - B first decrypts with its private key K-priv-B - only B can do this!
  - B then decrypts with A’s public key K-pub-A - only A could have sent the message!



- Another way to exchange keys is the Diffie-Hellman key exchange protocol
    - It was first published in 1976

- How does the Diffie-Hellman key exchange work?
    - Alice and Bob wish to agree on a shared secret key
    - First, they agree on two values “m” and “g”
    - “m” is a large prime number and (m-1)/2 must also be prime and “g” is an integer
    - Alice generates a large random number “a”, which is kept secret
    - Alice computes x = g^a mod m and sends the result to Bob
    - Bob generates a large random number “b”, which is kept secret
    - Bob computes y = g^b mod m and sends the result to Alice
    - Alice computes K1 = y^a mod m
    - Bob computes K2 = x^b mod m
    - Note that K1 = K2

- Here is a simplified example (with very small numbers)



- This process is vulnerable to a “man in the middle” attack
    - Suppose Trudy is in the middle of Alice and Bob
    - Alice thinks she is talking to Bob but is actually talking to Trudy and negotiating a key with him
    - Bob thinks he is talking to Alice but is actually talking to Trudy and also negotiates a key with him
    - Now Trudy can decrypt and encrypt all messages between Alice and Bob, without them knowing

Authentication Protocols

- We use Authentication Protocols to ensure that the other party in an exchange is who they claim to be
    - Challenge response protocols are widely used for this purpose
    - They work with both symmetric and asymmetric encryption (for symmetric encryption we assume that the key has been exchanged successfully)

- How does a challenge-response protocol work?
    - Assume there are two parties A and B
    - A sends an encrypted nonce (a truly randomly generated number that will only be used once) to B
    - B decrypts the message and makes some prearranged modification to the number, e.g. it will add 1 to the number
    - B encrypts the result and sends it back to A
    - Now A can trust B
    - For B to have confidence in A repeat the process in the other direction

Digital Signatures and Electronic Contracts

- A digital signature must be unforgeable
    - Nobody else can sign
- It must be authentic
    - Check if the signature is correct
- The signature makes sure that ...
    - … no alterations are possible to document
    - … no reuse is possible
- Asymmetric encryption is one solution (check section “Key Exchange” above for an example)
    - Usually the message digest is signed
    - This is also done in electronic contracts (see directly below)

- In an electronic contract between A and B both first agree on a hash function H
    - They both calculate a message digest MD = H(contract) to get MDa and MDb
    - A and B already have asymmetric key pairs so each now signs the digest with their own private key
    - SMDa = E(K-priv-a, MDa), SMDb = E(K-priv-b, MDb)
    - Exchange these signed digests, each now signs the other signed digest with their own private key
    - DSMDa = E(K-priv-a, SMDb), DSMDb = E(K-priv-b, SMDa)
    - Now append these two double-signed digests to the contract
- If a third party (e.g. a court of law) wishes to verify the contract they compute

    - TPMD1 = H(contract)
    - TPMD2 = D(K-pub-b, D(K-pub-a, DSMDa))
    - TPMD3 = D(K-pub-a, D(K-pub-b, DSMDb))
    - If TMPD1 = TMPD2 = TMPD3 then the contract is valid

X.509 Certificates and Certificate Authorities

- Who can you trust?
    - Maybe there is a respected individual you already know who introduces you to someone else
    - Therefore, you can now trust this other person as well
- The same idea is used to validate a public key
    - The key and the identity are bound together in a certificate, signed by the “respected individual”
    - A widely used mechanism are X.509 certificates

- In X.509 the “respected individual” is replaced by a certificate authority (CA) or certification authority
    - In return for a fee, the CA will verify your identity and issue you an X.509 certificate
    - This certificate ties together your identity with your public key (and the public key matches your private key)

- What does a X.509 certificate contain?
    - version number
    - serial number
    - validity period
    - unique identifier of the certificate holder
    - certificate holder public key algorithm
    - public key for the certificate holder
    - unique identifier of the certificate issuer
    - signature algorithm description
    - certificate signature
- Here is an example certificate of secure.ecs.soton.ac.uk

Common Name: secure.ecs.soton.ac.uk
Organization: University of Southampton Organization Unit: iSolutions
Locality: SOUTHAMPTON State: Hampshire
Country: GB
Valid From: April 23, 2019
Valid To: April 23, 2021
Issuer: QuoVadis Global SSL ICA G3
Subject public key info:
  key algorithm: RSA
  n=009788907bba1f89...
  e=010001
Signature algorithm: SHA256 with RSA
Signature: 393e65a573bef69c...
Serial Number: 1f682ec44954b2b944771628072f247b7d1d24ff

- How can you trust the CA
  - It works by creating a chain of trust
  - Eventually there must be a root CA that you have to trust
  - All browsers have self-signed certificates issued by the major CA's built into them
  - If one of these CA's is compromised it is a serious problem!
- Example of a root CA

Common Name: QuoVadis Root CA 3 G3
Organization: QuoVadis Limited
Country: BM
Valid From: January 12, 2012
Valid To: January 12, 2042
Issuer: QuoVadis Root CA 3 G3
Subject public key info:
  key algorithm: RSA
  n=00b3cb0e10678eea...
  e=010001
Signature algorithm: SHA256 with RSA
Signature: 3461d956b5128755...
Serial Number: 2ef59b0228a7db7affd5a3a9eebd03a0cf126a1d

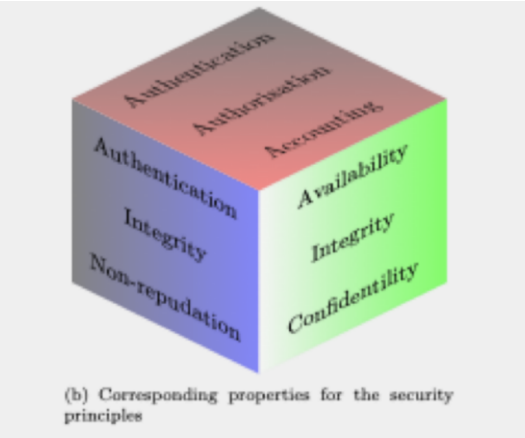
Nawful's Part

Authentication

- First, a definition of a few things which are commonly mixed up
- What is authentication?
    - It's checking a claim of identity
    - Example: Is the person who logs into the system really the person they claim to be?
    - Common problems are that sometimes you forget the password or you enter it incorrectly
- What is authenticity?
    - It's checking whether something is legit
    - Example: Is the email from my colleague really from him?
- What is integrity?
    - It's checking whether the data has not been altered/destroyed
    - Example: Is the email from my colleague unchanged?
- What is authorisation?
    - It's checking whether you (the user) are allowed to do what you are trying to do
    - Example: Are you allowed to access the admin panel?
    - We use access control mechanisms for authorisation
    - You can be authenticated to a system but not authorised to perform certain actions
- What is accounting?
    - It's tracing/logging every action that happens
    - Example: Every log-in attempt is logged
- What is non-repudiation?
    - The parties involved in a data exchange cannot deny the fact that they were involved
    - Example: The email came from the university because it is digitally signed with the private key of the university
- What is confidentiality?
    - It's making sure that only the intended people can read a message
    - Example: I send an email to a friend and encrypt it so that only they can read it because it contains sensitive information
- What is availability?
    - It's the guarantee of reliable access to information by authorised people
    - Example: I can access the shared cloud drive that contains sensitive documents reliably

Models

- There are different cyber security models
- CIA model: Three key principles which should be guaranteed in any kind of secure system
    - Confidentiality
    - Integrity
    - Availability
- AAA model: Three key principles for access to a secure system
    - Authentification
    - Authorisation
    - Accounting
- AIN model: I couldn't find anything about it online, but my guess is: Three key principles to ensure secure communication between parties
    - Authentification
    - Integrity
    - Non-repudiation



(b) Corresponding properties for the security principles

Types of Authentication

- How can we perform authentication?
- There are different types of stuff we can use
- **Knowledge**
    - Information or data which is known only to you
    - Examples: Passwords, PINs
    - Problems: Easy to hack if they are short, poorly selected, can be shared with other people, can be forgotten
- **Possession**
    - A physical object which belongs to you
    - Examples: ID cards, smart cards
    - Problems: You can forget them, the scanner does not work

- **Biometrics**
  - Your unique physiological and behavioural characteristics
  - Examples: Fingerprint, iris recognition, facial recognition
  - Problems: Hard to implement, expensive hardware necessary

- You can use additional factors to make the authentication stronger
- This is called multi factor authentication
  - The combination of multiple techniques improves the security
  - Two-factor authentication (2FA) is widely used: An attempt to logon first requires a password, the system then sends a text message you have to enter
  - Other factors are your geolocation or browser you are using

Authentication Systems

- Passwords: How to choose a good one?
  - Use all characters (not just a-z)
  - Choose long passwords with at least 10 characters
  - Don’t use actual names or words
  - Choose an unlikely password
  - Don’t tell anybody
- Single Sign-on (SSO)
  - The user authenticates once per session, the original system then authenticates user to all other systems
  - The strength of the SSO system determines the strength of the overall system
  - You need to trust whoever provides the SSO
  - Example: At work you log in once, then you can access all company services
- Time-based systems
  - A small portable unit that generates a one-time password on demand
  - Therefore, a possession mechanism
  - The password will be valid only for a short period of time
  - Needs time-synchronisation between the device and the server and must not be possible to reverse-engineer
- One-time passwords
  - You distribute a list of passwords to the users
  - Whenever a user logs in they use a password from the list
  - Each password can only be used once
  - It was used for remote banking systems to validate transactions
  - However, it is inconvenient to have long lists of passwords and ensuring that the list is safe and secret

Passwords

- How do you save passwords securely? We will hash all passwords and store those hashes!
- How does the hash function work?
    - Basically, this was already discussed in Ed's part
    - The idea is that you have a one-way hash function (can't go backwards) to produce a hash from the plaintext password
    - The function is designed so that it is computationally intensive to find the plaintext from a given hashed value - you'd have to use brute force to find out
- How do you verify hashed passwords?
    - When the user provides a password for log-in we compute the hash from the password provided and compare it to the hash stored
    - If both match, we grant access
- When we hash passwords, we also add a "salt"
    - This is a random number which we use when we hash the password
    - We also store this random number (together with the hash)
    - It has the effect that two identical passwords result in two different hashes - provided that the salt used is different
    - It protects against searching for hashes against rainbow tables of hashed passwords
    - A rainbow table contains mappings from plaintext passwords to their hashes (which were hashed without salt)
- What are some threats and bad practices?
    - Users can be persuaded to tell you the password
    - Users often write passwords down
    - Users re-use passwords for different systems
    - Users use simple passwords
    - Systems store passwords as plaintext
    - Backups have lower security measures

- What are some typical password attacks?
    - Brute force: Try every possible combination
    - Null authentication: Can you log-in by providing no password at all?
    - Username and password are the same: This one is obvious
    - Social engineering: What are the most likely passwords for a user? The name of their child, brother, sister, pet? Or you just call the person and ask to verify their password as a "security check"
    - Dictionary attacks: Try out common words
    - Common passwords list: Try out common passwords
- The following table shows how long it takes to brute force an 8/10 character password made up from upper & lower case letters and digits

Scenario	Online attack	Offline attack	Massive cracking array scenario
Description	The attacker has to send requests to a	The attacker has the hashed password	The attacker has the hashed password

	server to check the password attempt	locally	locally and uses supercomputers
Number of guesses	One thousand guesses per second	One hundred billion guesses per second	One hundred trillion guesses per second
8 character password	70.56 centuries	36.99 minutes	2.22 seconds
10 character password	2.71 hundred thousand centuries	3.25 months	2.37 hours

Physical Biometrics

- There are different physical biometrics we can use
- **Fingerprint**
  - A fingerprint is made up of ridges and furrows located on the fingertips
  - The uniqueness of a fingerprint can be determined by the pattern of ridges and furrows
- **Facial**
  - Facial recognition is an automated method to record the spatial geometry of the distinguishing features of the face
  - The features can be based on the location and shape of facial attributes (e.g. eyes, eyebrows, nose, lips, chin) and their spatial relationships
- **Retina**
  - Measuring the blood vessel patterns in the retina at the back of the eyes
  - Unique for every individual
- **Iris Scan**
  - Each iris is unique: right and left eyes are different
  - Complex patterns with many distinctive features such as furrows, ridges, crypts, rings, coronas, freckles and zigzags

- How do we acquire physical biometrics?
  - Through a live scan of the biometric feature by a biometric scanner
  - The acquired biometric image is used by a feature extraction module to compute the feature values corresponding to the position and orientation of certain critical points
  - These points are known as minutiae points
  - In the matching stage, the minutiae patterns extracted from the user's biometrics are compared with those in the template

- How can physical biometrics be attacked?
  - Using masterprints
  - Harvesting unsecured images
  - Harvesting organs

    - Using forged biometrics
    - Exploiting software vulnerabilities
    - Reusing residual biometrics

Behavioral Biometrics

- There are different behavioral biometrics we can use
- **Signature verification**
    - An automated method of examining the way an individual signs his or her name
    - Data collected using a stylus (pen) and a sensor pad
- **Voice recognition**
    - The sound of a human voice is primarily based on physical characteristics such as vocal tracts, palate, teeth, nasal cavities, lips and mouth
    - The physical characteristics of human speech are invariant for an individual, but learned behavioral patterns change over time because of age, medical conditions and emotional state
- **Keystroke dynamics**
    - Measures dynamics such as typing speed and pressure, the total time of typing a particular password, and the time a user takes between hitting certain keys

- How can behavioral biometrics be attacked?
    - Human beings are predictable
    - Signatures can be forged
    - Voice synthesisers can be trained
    - Exploiting software vulnerabilities

How Biometrics Work

- Biometrics can be used for both Authentication and Identification
    - Identification is about finding out who the person is
    - The identity of the person is unknown until we search the database
    - Many matches must be searched
    - Ideally, we need a database containing everybody

- What functions are provided?
    - Enrolment function: Acquire a user's biometric raw data, create a biometric template and save it
    - Authentication function: Acquire a user's biometric raw data, create a biometric template and compare it to the stored template

    - Matching score function: A matching score quantifies the similarity between the input and the stored template. The higher the score, the greater the confidence that the samples belong to the same individual

- How is the accuracy of matching expressed?
    - There are two measures: False Match Rate (FMR) and False Non-Match Rate (FNMR)
    - The FMR is the rate of individuals who are matched even though they are not the same - also called false accept rate (FAR) or false positive
    - The FNMR is the rate of individuals who are not matched even though they are the same - also called false reject rate (FRR) or false negative
    - In a perfect system both FMR and FNMR are 0 - but no biometric system is perfect
    - If you increase the threshold of the match scoring function you will make the system more secure but increase the FNMR
    - If you decrease the threshold you will make the system more tolerant to input variations but the FMR will increase

Biometric Issues

- The seven key biometric issues are
    - **Universality:** All human beings have the same physical characteristics, e.g. fingers, face, iris, etc.
    - **Uniqueness:** For each person their physical characteristics are distinct
    - **Permanence:** These characteristics remain largely unchanged over time
    - **Collectability**: A person's unique physical characteristics need to be measurable and able to be collected reasonably easily
    - **Performance:** The identification accuracy must be acceptable before the system can be made operational. Other performance metrics include speed and resource requirements
    - **Acceptability:** The user population and the public in general should have no (strong) objections to the measuring/collection of the biometric data
    - **Resistance to circumvention:** In order to provide acceptable security, the system needs to be robust and to withstand fraudulent attacks

- Performance issues
    - The accuracy, retrieval and storage must be considered
    - Two samples of the same biometric characteristic from the same individual will not be exactly the same because of imperfect measurements, changes in user characteristics, user interaction with sensor, noise, etc.

- Privacy issues
    - Biometric data could be used for purposes for which it was not intended, e.g. medical diagnosis

    - Subjects could be identified when they didn't want to be (without consent)
    - Governments build large databases - do we trust them?

- Security issues
    - Is the identity of the person enrolled correct?
    - Is the enrolment process secure?
    - Is the database storage secure?
    - Is the template transmission secure?
    - Is the authentication process secure?
    - What about the accuracy of recording and matching?

Threats to Authentication Procedures and Data

- What confidence tricks are there?
    - Phishing emails: Victim clicks on a fake email and installs malware or enters password
    - Telephone phishing: "Hello, this is Microsoft support, please tell me your password so I can fix your PC"
    - Person-to-person phishing: The attacker asks the victim in person or uses physical attacks
    - Typographic attacks: Spoof a URL, e.g. paypa1.com looks almost like paypal.com

- What are some remote technical tricks?
    - Spoof techniques: Those are used in phishing emails. E.g. the attacker sends a link to a fake site that looks like the real deal
    - Cross frame scripting: The attacker embeds another page in an iFrame and can then spy on the victim
    - Infected software: Well, malware basically
    - DNS poisoning: Infecting DNS servers so they answer with the attackers IP instead of the real one
    - Traffic sniffing: Capture traffic in your network, can read all the unencrypted stuff
    - Exploiting outdated technology: Make use of well-known security vulnerabilities

- What are some local technical tricks?
    - Software vulnerabilities: Again, exploiting known security vulnerabilities
    - Browser "toolbars"/extensions: They might have unrestricted access to what you enter on a page
    - Trojans: Covered in Ed's part
    - Viruses: Also in Ed's part
    - Authorised exploitation: Some authorities mistakenly thought that some malware was safe, so the victim believes it is safe

    - Visual tricks: Spoofing the URL or hiding the address bar in a browser
    - Hardware attacks: Attaching a physical devices such a keylogger device

- What are some common mistakes that victims make?
    - Writing down passwords
    - Telling people passwords
    - Picking weak passwords
    - Inattentiveness when entering passwords: Entering them on a fake website, not checking whether someone is watching them, etc.
    - Permitting accounts to be “borrowed”
    - Physical attack: Getting mugged
    - Allowing weak procedures for when they need to recover the password
    - Using outdated technology

- Here are the common implementation oversights
    - Unsafe lost password procedure
    - Insecure cookies
    - Trusting form data without validation
    - Accepting sensitive info over non-SSL
    - No brute force protection
    - Not blocking logins from certain locations (this does not apply always)
    - Allowing users to save login data in their browser
    - XSS vulnerabilities
    - No input data sanitisation
    - No output data sanitisation
    - Cryptographic oversights - e.g. using pseudo-random numbers
    - Not encrypting sensitive data

- Sometimes users get blocked/locked out of the system
    - DOS (denial of service) attacks can be used to perform failed logins, which result in the user being locked out
    - They can also be used to acquire the user to reset their password
    - Enrolment attacks can be carried out by a deliberate wrongdoer who creates a new set of credentials
    - Or they can be carried out by identity squatters who register your name/nickname/persona before you do

- What can compromised authentication lead to?
    - Interception: The attacker can intercept the messages. This is an attack on confidentiality
    - Interruption: The attacker can make the system unavailable. This is an attack on availability
    - Modification: The attacker can alter the messages. This is an attack on integrity
    - Fabrication: The attacker can make up messages. This is an attack on authenticity

Morris and Thompson paper

- Morris and Thompson suggest the following things
    - Have a solid authentication system.
    - Encrypt passwords.
    - Make sure your local/remote users database is secure.
    - Make sure its resilient to dictionary attacks.
    - Varied character types (Lower Case, Upper Case, Numbers & special characters).
    - Use password manager suggestions. (might be destructive if forced on the users.)
    - Use proper encryption and Key size.
    - Chose a long password.
    - Use Salted passwords
    - Don't feedback if username is correct
    - Use a computationally expensive encryption.

Encryption Guidelines

- Which encryption should be used for which case?
- Check the following table

Personal identity verification Key Type	Algorithms and Key Sizes
Personal identity verification Authentication key	RSA (2048 bits) ECDSA (Curve P-256)
Asymmetric Card Authentication key	RSA (2048 bits) ECDSA (Curve P-256)
Symmetric Card Authentication key	3TDEA3 AES-128, AES-192, or AES-256
Digital signature key	RSA (2048 bits) ECDSA (Curve P-256 or P-384)
Key management key	RSA key transport (2048 bits); ECDH (Curve P-256 or P-384)
Personal identity verification Secure Messaging key	ECDH (Curve P-256 or P-384)

Pervasive Monitoring (PM): From Bad to Worse

Definition and Introduction

- What is pervasive monitoring?
    - It's basically monitoring people constantly
    - It can be considered as an attack
- It's good for some applications, e.g. medical purposes
    - However, what if that was compromised?
    - What if it's also used for malicious purposes?

- Pervasive Monitoring can be an attack or can be used as part of an attack
    - It is not the only security or privacy issue we need to work on (Spam, Malware, DDoS, …)
    - However, mitigations for Pervasive Monitoring can also help a lot with other problems
- Pervasive monitoring targets everyone and at a massive scale
    - If there is more than one way to carry out PM they will try all of them
    - Data is collected even offensively

- What does it take to justify pervasive monitoring?
    - 1 tea spoon of terrorism
    - 2 tablespoons of fear mongering
    - ½ tea spoon trespass on civil liberties
    - 3 or 4 politicians
    - 2 pounds of the press
    - A small dash of public approval

Examples

- For example, GCHQ (UK intelligence agency) intercepted millions of Yahoo webcam images
    - With aid from the NSA they intercepted and stored the webcam images of millions of users
    - Users were not suspected of wrongdoing
    - This was carried out under the surveillance program was codenamed Optic Nerve
    - The program collected still images of Yahoo webcam chats in bulk and saved them to agency databases, regardless of whether individual users were an intelligence target or not

- Example: GCHQ faces new Belgacom (telecommunications company from Belgium) hack allegations
    - Leaks from NSA whistle-blower Edward Snowden reveal that the alleged GCHQ cyber-attack on Belgacom used Regin malware
    - It was undiscovered for two years before it was detected
    - There are new concerns that the clean-up operations was not successful
    - Snowden revealed in 2013 that Belgacom had been hit by an advanced persistent threat (APT) attack - more about APT later
    - He says this was the work of the UK's GCHQ

- Example: State Funded Firmware Hack "Sonic Screwdriver & LoJax"
    - The CIA apparently developed an implant for Apple's computers that used the Extensible Firmware Interface (predecessor of UEFI)
    - It required physical access to the targeted computer and a malicious Thunderbolt Ethernet adapter called the "Sonic Screwdriver"
    - LoJax, however, is an entirely different animal
    - It was built to be deployed remotely
    - It can read and overwrite parts of the UEFI firmware's flash memory

Motivation

- I actually don't know why the title of this section is "Motivation" (it's from the slides)
- Here are the names of some countries that are watching you…
    - The '5-Eyes': The US, the UK, Canada, New Zealand and Australia
    - The '9-Eyes': The '5-Eyes' group plus Denmark, Norway, the Netherlands and France
    - The '14-Eyes': The two above groups plus Germany, Sweden, Belgium, Spain, and Italy

- What are the different branches of intelligence?
    - Signals intelligence (SIGINT)
    - Defence intelligence (DEFINT)
    - Human intelligence (HUMINT)
    - Geospatial intelligence (GEOINT)

Pervasive Monitoring and other Stuff

- PM vs. targeted monitoring
    - Most people are against pervasive monitoring, but all right with targeted surveillance (e.g. under warrant wiretap)
    - This is also often called "lawful interception"
    - However, lawful interception can be, and has been used for, pervasive monitoring

- PM vs. pervasive advertising
    - Data driven advertising will always be a breach of your privacy
    - It will collect even more data than government agencies do!
    - Government agencies can convince or coerce companies into collaborating
    - Google's core business is around advertising so how can we convince internet business not to use ad-trackers?
    - How to balance toxicity of data vs. data- mining benefits?

- PM and data mining
    - "The problem that we face today is not data, or how to acquire it. The Problem is what it means."
- The Facebook–Cambridge Analytica data scandal was a major political scandal in early 2018
    - It was revealed that Cambridge Analytica had harvested the personal data of millions of people's Facebook profiles
    - They didn't have their consent and used it for political advertising purposes

Defense

- What can we do against PM?

- RFC 7258/BCP 188 state that PM will be considered as an attack in the design process of all protocols
    - Furthermore, it states that IETF (Internet Engineering Task Force) will work on introducing new and revising existing protocols to mitigate PM by design
- RFC 7435 defines "Opportunistic Security"
    - Any system with OS will try to encrypt all communication when talking to other systems
    - If this doesn't work, it has to fall back to cleartext communication

- IAB (Internet Architecture Board) recommends to
    - Design for confidentiality by default - encrypt everything
    - Create a PM threat model document

- DNS Privacy: DNSSEC does not encrypt the payloads of DNS queries or responses but provides a method of validating the authenticity of the information

- Some other non-technical solutions are
    - User education
    - Free and open source software
    - Legislation (e.g. GDPR)
    - Clever cryptography

Why it will not go away

- Government or corporate espionage is not going away
- Privacy invasive commerce (legitimate and not) is very lucrative
- Lack of legal accountability mechanisms
- Its not well understood so it keeps getting worse which leads to badly-informed decision makers
- Slow Government regulation of businesses (e.g. Data Protection Agencies)
- Slow Commercial reaction to user privacy requirements
- Low attention to NGOs working to enhance privacy.
- Technical privacy enhancing/enforcement mechanisms are still in their infancy

Consequences

- Tighter export control on information
    - News blackout on revolution
- Censorship
    - Arrests and assignations of activists
- More power to authoritarian regimes
    - Corruption and wasted public funds
- Network Neutrality
    - VOIP applications like WhatsApp and Viber might get blocked
    - Sometimes internet access might be restricted
- Hate Speech
    - There's plenty of it around

Advanced Persistent Threats (APT)

Definition

- What is an Advanced Persistent Threat?
    - The FBI getting around measures you took to be anonymous, focusing on your identity and physical location
    - The CIA, suspecting you to be a spy, breaking into your hotel room and installing spyware on the laptop that you thought was secure in the hotel safe
- What is not an Advanced Persistent Threat?
    - Your housemate posting "I love cybersecurity" using your Facebook account
    - A script-kiddie uploading a bunch of malicious scripts, with little to no comprehension of what they accomplish
    - A hacktivist group that regularly DDoS websites of organisations they don't like

    - Random hackers who spam a pre-scripted, highly platform-dependent attack at government organisations
- As you can see, APT is some pretty serious shit

- Let's breakdown APT
- Advanced
    - Attacker adapts to the victim's efforts
    - Attacker can develop or buy zero-day exploits
    - Attacks are highly sophisticated
- Persistent
    - Attacks are objective and specific
    - Attacks will continue until the goal is reached
    - Attacks are intended to maintain long term connectivity
- Threats
    - The threat is not just the malware/exploit/attack
    - This is more about the entities behind the attack

- The common methods used are (more about them later)
    - Kill Chains
    - Diamonds
    - Mitre Att&ck
- Tools used are
    - Zero-day exploits (explained later)
    - Rootkits

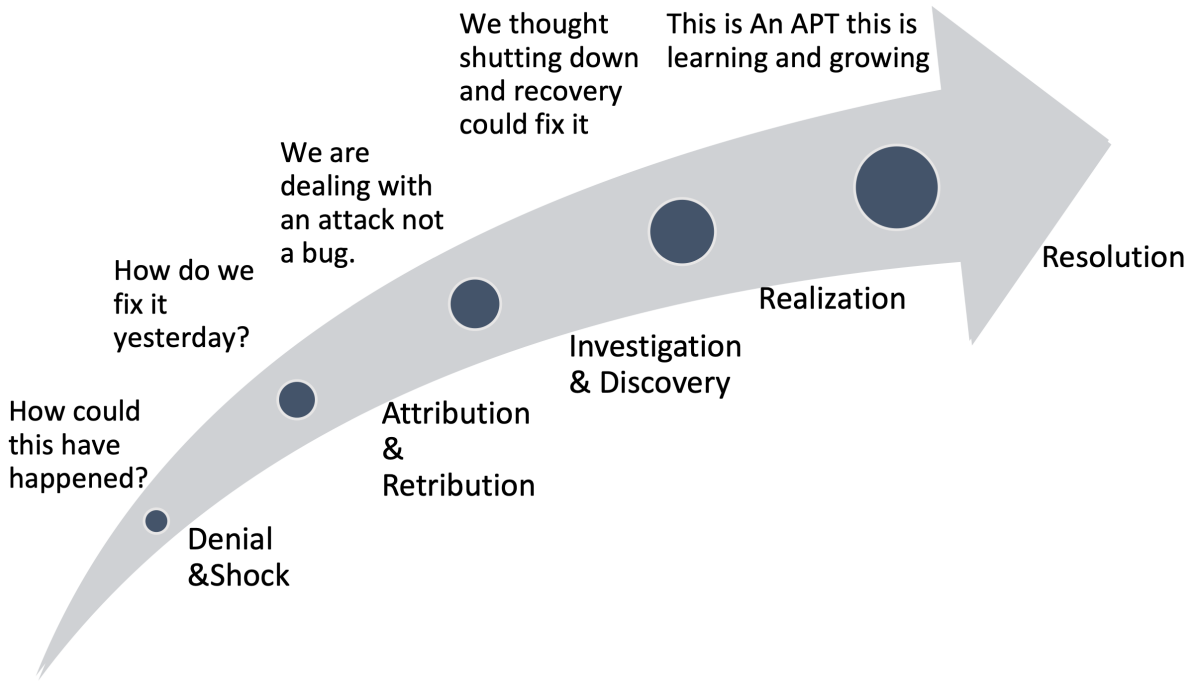
Sponsors & Attackers

- Who sponsors APT and who attacks?
- Potential sponsors are
    - Commercial companies (e.g. competing company)
    - Military
    - Intelligence agencies
- Attackers are both well funded and highly skilled

- What is the motivation behind APT?
    - Espionage
    - Sabotage
    - Brand Damage
    - Attacker is late to the market (At least I think that's what the slides mean)
    - Counterfeiting

Discovering APT

- Here is a cool graphic that shows the typical stages people go through when they discover the APT



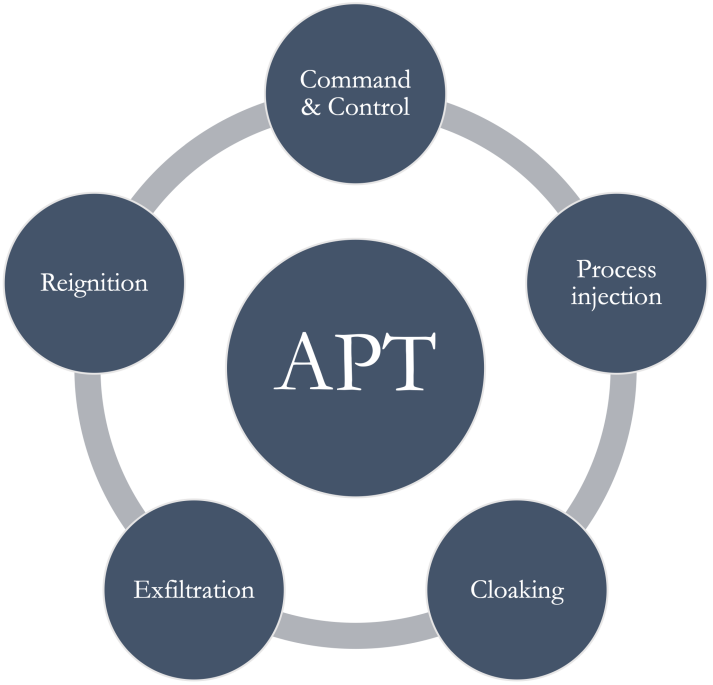
Technical & Behavioral Characteristics

- APT works everywhere
  - Linux
  - Windows
  - OS X
  - Firmware
- What are the main technical characteristics?
  - It's customised code and not a common code module
  - It's focused on objectives and not opportunistic
  - It may use multiple zero-day exploits
  - The deployment is controlled by human intervention and not fully automated
  - It operates in low and slow manner to remain stealthy and unnoticed

- What are the main behavioral characteristics?
  - It's directed at political and military targets
  - It uses multiple vectors of attack
  - It causes low to zero downtime to systems
  - It operates very stealthy
  - It's persistent

Anatomy

- Here is a very exciting picture showing the different parts of APT



- What is command & control?
    - The ability for the remote attacker to direct tasking and configuration of the implanted malware, to download new payloads and to provide malware updates
- What is process injection?
    - The most sophisticated APTs don't operate as applications or processes, but attach themselves to an existing application or process that's running in memory
- What is cloaking?
    - An APT wants to remain invisible for as long as possible, and operates as a low and slow attack, stealthily extracting what it needs, with as little impact on the host computer, and without generating regular or predictable network traffic
- What is exfiltration?
    - APT is designed to collect information, and it needs to send it back to its control server
    - A good Exfiltration system will not only encrypt the information being sent, so that it isn't seen by any monitoring systems, but it may also hide it in the kind of packets that are normally ignored, such as HTTP or DNS requests
- What is reignition?

    - In order to remain operational for a period of time, an APT needs to restart when the system is rebooted, or if the system administrator attempts to remove it
- What is love?

Zero-day Vulnerability

- What are software vulnerabilities?
    - Jk, I'm fairly sure you know what they are by now

- When you find a vulnerability, you should …
    - Notify the vendor of the vulnerability
    - Provide the vendor a reasonable amount of time to fix it
    - Disclose the vulnerability publicly (a while after the fix has been released)
- What if the vulnerability is not disclosed?
    - Then we can call this vulnerability a zero-day vulnerability
    - This is a vulnerability that has been discovered but the vendor has not been informed or has not found it yet, and it has not been published

- Zero-days are dangerous because
    - We don't know about them so we might fail to detect attacks that make use of them
    - We don't have patches for them

- How can we discover zero-days?
    - We can set up a honeypot or honeynet - that is a fake system that appears to be legit to the attacker and when it gets compromised we know there is an attack
    - We can closely monitor the hacking community on the dark web - zero-days are often advertised/discussed there
    - We can use testing frameworks and debug tools
    - We can carry out firmware analysis and stress testing

- How do attackers exploit zero-days?
    - Obviously, you have to know about a zero-day vulnerability to exploit it
    - You need to have the skills
    - You need to have the tools
- The window of vulnerability is the time between the discovery of a zero-day vulnerability and the release of the security update

Discovery & Defending APT

- How and when are APT found?

    - To find an APT focus on each stage of the kill chain (more about it later), this will provide the opportunity for early detection
    - APTs are usually found when network monitoring detects the installed malware attempting to connect to its command & control server
    - Nevertheless, the average time it takes to detect an APT is measured in months

- How to defend against APT?
    - Build a strong security foundation
    - Implement strong encryption
    - Use rigorous monitoring
    - Maintain proper configuration
    - Regularly patch and update and check what has been fixed in updates
    - Have a good defense to at least delay the attacker
    - Maintain proper backups

- You will still fail to defend against APTs because
    - You don't have knowledge about zero-day vulnerabilities
    - You are not well funded - compared to the attackers
    - You are not well organised - compared to the attackers

Use Case - Stuxnet

- Stuxnet was an APT detected in 2010
    - It was designed specifically to target centrifuges in the Iranian nuclear program
    - It was targeting the Siemens industrial plant equipment used in nuclear fuel enrichment which was used in the uranium enrichment facility at Natanz, Iran
    - The US admitted in 2012 that it was responsible, together with Israel, for developing Stuxnet

- Stuxnet had its own rootkit
- It had 7 means of communication/propagation
    - Peer-to-peer communication and updates
    - Infecting WinCC (Siemens system) machines via a hardcoded database server password
    - Propagating through network shares
    - Propagating through the print spooler
    - Propagating through the Windows server service
    - USB stick
    - Project files
- It recorded all data and used it for machine learning
- It was very stealthy and activated only once every 27 days
- It worked offline

- It made use of 4 previously unknown zero-day vulnerabilities
- It targeted the Siemens SCADA (supervisory control and data acquisition) system

- When Stuxnet was active it repeatedly sped up and slowed down the centrifuges, causing the aluminium tubes to expand and contract, eventually destroying between 900 and 1000 centrifuges

- Here's a TED talk if you want to learn more stuff about Stuxnet

Stages of Attacks, Killchain and Mitre Att&ck

- An active cyber attack can be divided into 5 different stages (simplified version)
- There also exists a so called kill chain that denotes 7 different stages
- Finally, there's also 12 stages of "Mitre Att&ck"
- These three models can be mapped to each other as shown in the following table

<table>
<tr><th>Active Attacks</th><th>Kill Chains</th><th>Mitre Att&ck</th></tr>
<tr><td>1 Reconnaissance</td><td rowspan="2">1 Reconnaissance</td><td rowspan="3"></td></tr>
<tr><td>2 Scanning</td></tr>
<tr><td></td><td>2 Weaponization</td></tr>
<tr><td>3 Gaining Access</td><td>3 Delivery</td><td>1 Initial Access</td></tr>
<tr><td>4 Maintaining Access</td><td>4 Exploitation</td><td>2 Execution</td></tr>
<tr><td rowspan="10"></td><td>5 Installation</td><td>3 Persistence</td></tr>
<tr><td rowspan="5"></td><td>4 Privilege Escalation</td></tr>
<tr><td>5 Defence Evasion</td></tr>
<tr><td>6 Credential Access</td></tr>
<tr><td>7 Discovery</td></tr>
<tr><td>8 Lateral Movement</td></tr>
<tr><td>9 Collection</td></tr>
<tr><td>6 Command and Control</td><td>10 Command and Control</td></tr>
<tr><td></td><td>11 Exfiltration</td></tr>
<tr><td>5 Clearing Tracks</td><td>7 Actions on Objectives</td><td>12 Impact</td></tr>
</table>

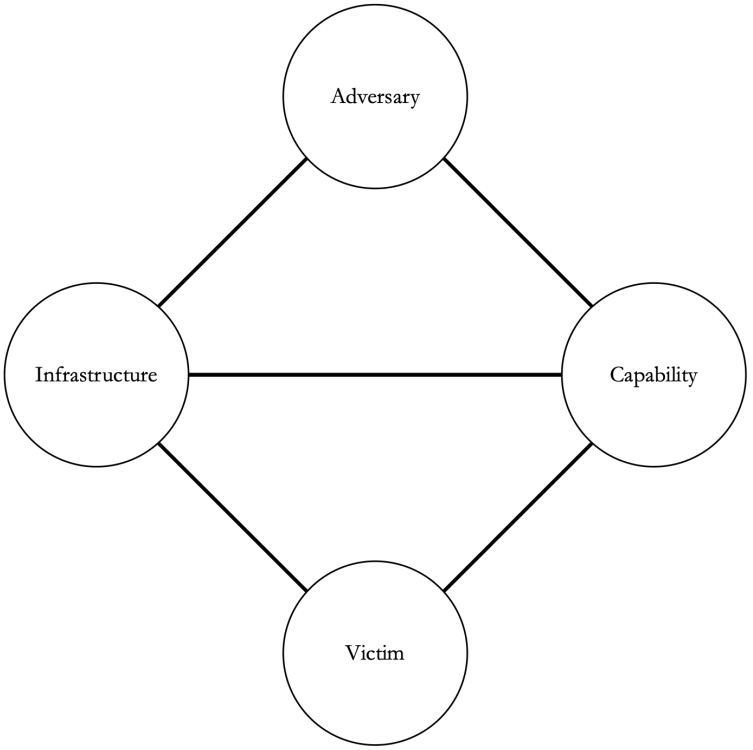
- You are asking yourself why these models can be mapped like this?

    - I don't know. I just write the notes.

Diamond Model

- We got yet another model, wohooo 🥳
- The diamond model is an intelligence-centric paper that maps
    - Tactics
    - Techniques
    - Procedures
- It focuses much more on understanding the attacker
    - What tools do they use?
    - What infrastructure do they have?
    - What are their motivations?

- The model looks like this:



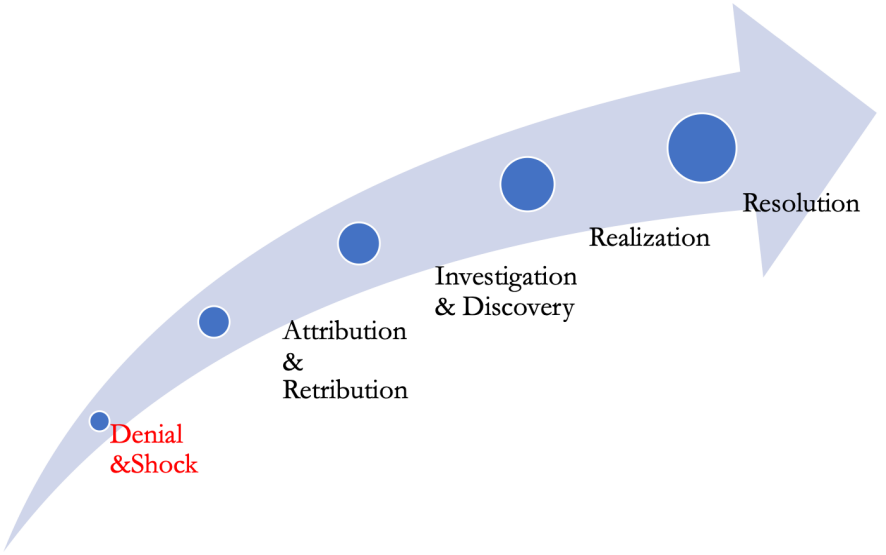
Use Case

- Alice, an employee, calls you to say that she is having problems opening a document from the new head of finance
    - You wander over, sure it's a PEBKAC error ("Problem exists between keyboard and chair")
    - You ask her to demonstrate the problem
    - She double clicks the attachment of the email

    - You see a long pause and then a blank PDF document opens

- You realise that something is wrong
    - You ask alice to open the email again and you see that the domain of the email address is wrong
    - Also, the message is rather brief, just “see attached”
    - You open task manager and see that Internet Explorer is running, however, no window is open

- You reach around the back of the PC to pull the network cable, preparing for the worst
- Congrats, you’ve just reached the first stage of discovering an APT - Denial & Shock



- We map the email to the diamond model & the killchain
    - The email is the delivery stage in the killchain
    - To map to the diamond model, the yellow highlighted bits is infrastructure and the green highlighted bits is capability

- Received: from [74.208.4.200] ([74.208.4.200:52006] helo=mout.gmx.net) by mail19.prod (envelope-from <mail-linkedin@gmx.com>) (ecelerity 3.4.2.33817r(MessageSystems/Momo-dev:9fc2f41b555d)) with ESMTPS (cipher=AES128-SHA) id89/31-18381-4E5D6A25; Tue, 10 Dec 2013 08:50:45 +0000
- Received: from abcnew.tk ([192.0.2.42]) by mail.gmx.com (mrgmxus002) with ESMTPA (Nemesis) id 0Lao5G-1VAPgv3HIW-00kOPw for <[Redacted]>; Tue, 10 Dec 2013 09:50:41 +0100
- From: Linkedin Security Team <alice.smith@example.net>
- To: REDACTED
- Subject: [ATTENTION]Mail System Update Confirm
- Thread-Topic: [ATTENTION]Mail System Update Confirm
- Thread-Index: AQHO9YTqWe+ROT7bGU6e8cQCiJMg0Q==
- Sender: "mail-linkedin@gmx.com" <mail-linkedin@gmx.com>
- Date: Tue, 10 Dec 2013 00:50:49 -0800
- Message-ID: <0ca9b59ffb2975dfadfc33b9a5455b05@abcnew.tk>
- Reply-To: Linkedin Security Team <hostmaster@mail-linkedin.com>
- Content-Language: en-US
- X-MS-Exchange-Organization-AuthAs: Anonymous
- X-MS-Exchange-Organization-AuthSource: [Redacted]
- X-MS-Has-Attach:
- X-MS-TNEF-Correlator:
- X-Mailer: mlx 5.1.7
- Content-Type: multipart/alternative; boundary="_000_0ca9b59ffb2975dfadfc33b9a5455b05abcnewtk_" MIME-Version: 1.0

- Moving on to the exploit/install stages of the killchain:
  - Link in the body of the email = http[:]//mail- linkedin.tk/mail/auth.php.
  - File named as runme.exe, with an MD5 hash of 4D0515C5F7985DA9759514AB0DE713915

- Using what we have seen so far, we can create a table including the delivery, exploit and install stages of the kill chain

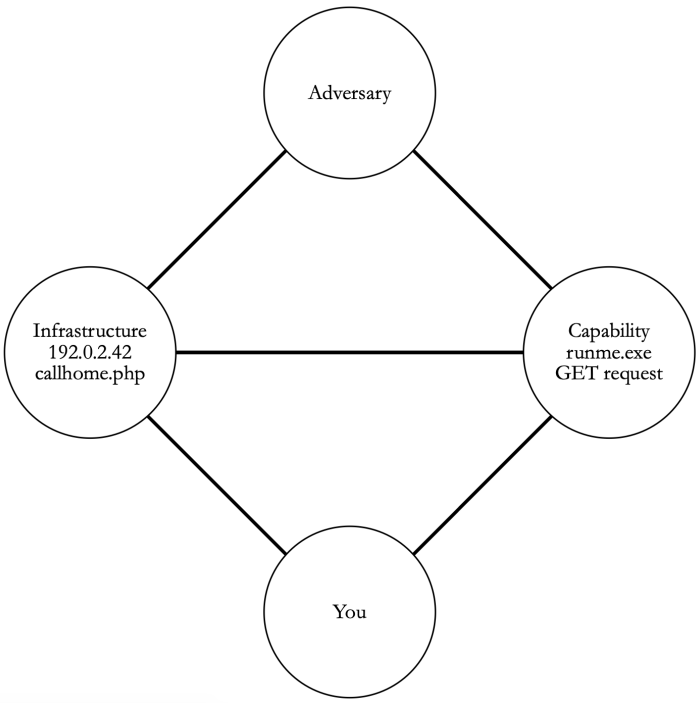
Item	Kill Chain	Diamond
mail-linkedin@gmx.com	Delivery	Infrastructure

98.126.223.106	Delivery	Infrastructure
Thread-Index: [Value]	Delivery	Capability
Date: Tue, 10 Dec 2013 00:50:49 -0800	Delivery	Capability
hostmaster@mail-linkedin.com	Delivery	Infrastructure
X-Mailer: mlx 5.1.7	Delivery	Capability
Boundary: [Value]	Delivery	Capability
http://mail-linkedin.tk/mail/auth.php	Exploit	Infrastructure
Payload.exe	Install	Capability

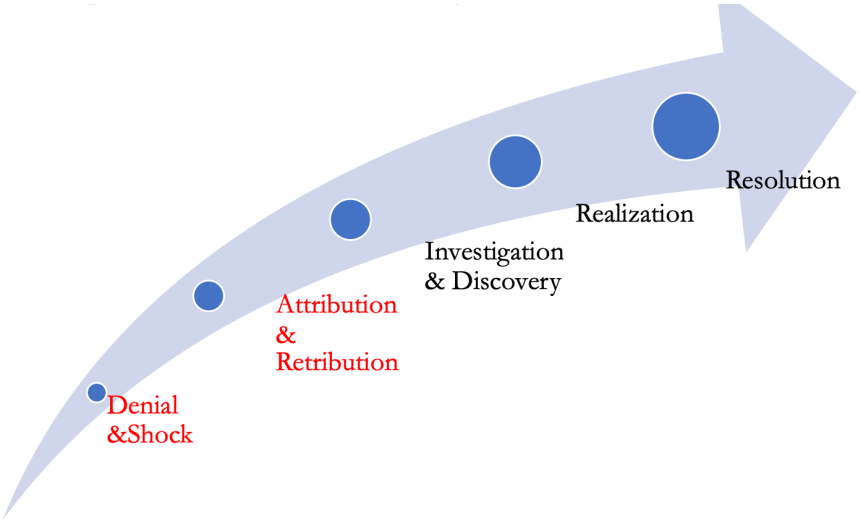
- In addition, we will gather more information after checking the network logs and the PC, which we add to the following table

Item	Kill Chain	Diamond
Email address: alice.smith@example.net	Delivery	Infrastructure
Sender IP: 192.0.2.42	Delivery	Infrastructure
Attachment: runmex.exe	Exploit	Capability
Network traffic: GET /callhome.php	Install	Capability

- We can also visualise this information graphically (using the diamond model)

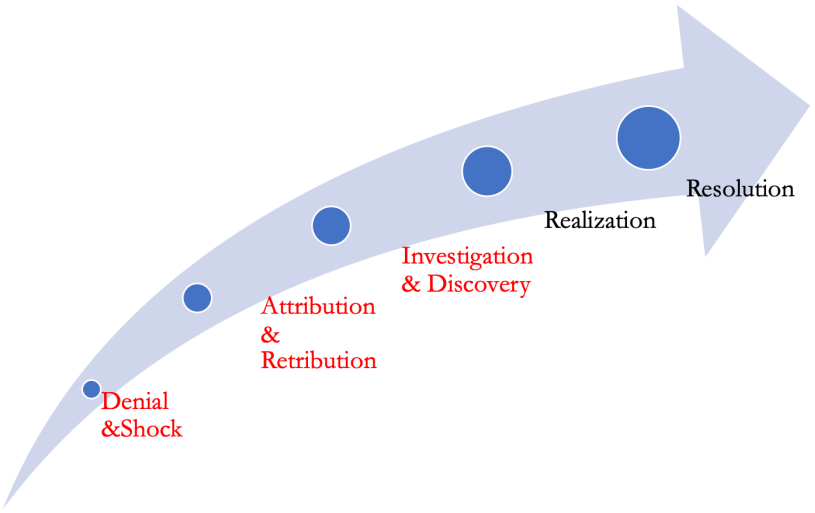


- Now we reach the second stage of discovering the ATP: Attribution & Retribution



- You create an IDS (intrusion detection system) signature for the network traffic, ensuring that the relevant IP addresses, email addresses and domains are blocked
- You submit the attachment to the antivirus provider
- You send out an email warning people that the company is being targeted by spear phishing, with pointers to elearning

- We move on to the third stage: Investigation & Discovery



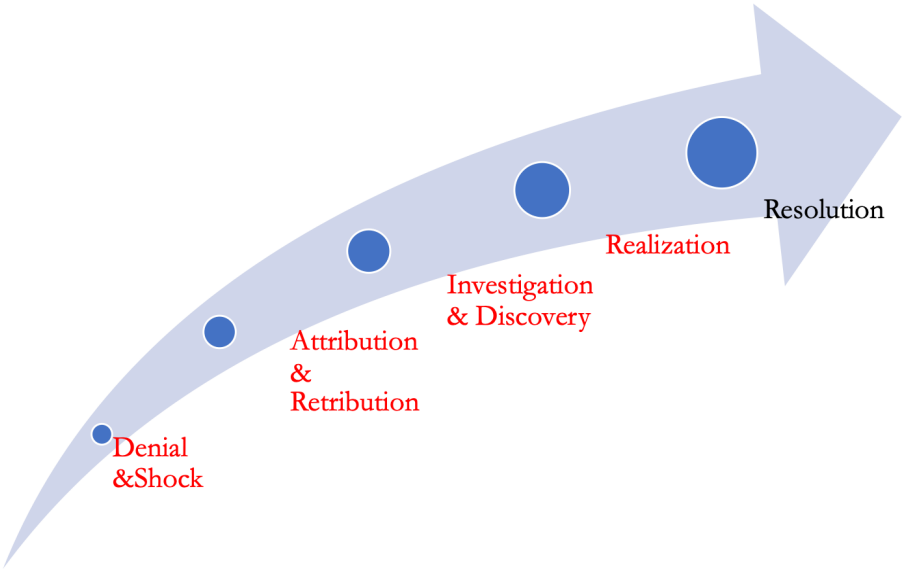
- After reviewing the logs of the mail server and firewall you see a few more attempts to deliver other attacks
- We can create another table with these new findings

Item	Kill Chain	Diamond
GET /aotj/581/link.php?a=5&z=SGVsbG8gV29ybGQ&f=476f6f6462796500 HTTP/1.1	C&C	Capability

Host: mail-linkedin.tk	C&C	Infrastructure
User-agent: Mozilla 4.4	C&C	Infrastructure

- Here is a summary of results after you have finished investigation & discovery
  - You have some quarantined emails from other IP addresses, so you added those IP addresses to the block lists
  - You updated the IDS models
  - You tested your mail server
  - You ran the attachments in a sandbox and checked the network traffic
  - You searched the mail server's logs to see if there has been any previous attack from the same IP addresses
  - You found similar emails containing malware and quarantined them

- You come back the next morning and reach the Realisation stage



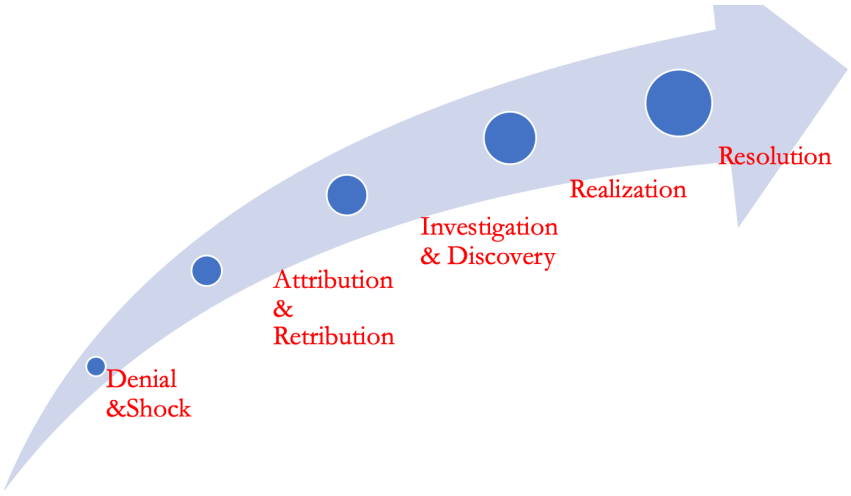
- You examine what your IDS has logged overnight
  - You find another email
  - You find another attachment (with the same malware)
  - You find new domains and IP addresses associated with the malware
  - You find that the attack started at 07:50 - before Alice got into the office
  - You find common header artefacts from the emails
- You put these new findings into another table

Item	**Kill Chain**	**Diamond**
Email address: <Alice.smith@example.com>	Delivery	Infrastructure
Email headers: X-Mailer: enom1856 & X-Campaign:	Delivery	Capability
Sender IP: 2001:0DB8:1de:ad00::1	Delivery	Infrastructure

Attachment: update.pdf.exe	Exploit / Install	Capability
Network traffic GET /callhome.php HTTP/1.0 POST /callhome.php HTTP/1.1	C&C Act on Objectives	Capability
Domains and IPs acme.example.net 198.51.100.5 2001:0DB8:1d3:ad00::5	C&C	Infrastructure
Destination IP for FTP 198.51.100.6	Act on Objectives	Infrastructure

- What you realise is
  - There is a human driving the attack
  - The attackers are persistent
  - The attackers work office hours
  - The attackers only used one tool so far
  - The attackers aren't risking their better-quality tools until they know their current tools won't work
  - The e-learning package on spotting phishing attacks isn't working
  - The attackers have exfiltrated a large volume of data already
  - The attackers have done their research on the target before they started the attack

- Moving to the final stage "Resolution"



- You find that
  - The group operating the attack is called BASILISK GAZE, a relatively new and immature group operating in the UTC+06:00 timezone
  - The targets have all been in the energy sector or supporting sector
  - The attack method is always an attachment in email and so far, it's always executables

    - Only 2 implants have been observed, the other being Poison Ivy with the default password
    - All observed exfiltration has been over FTP to an IP in the netblock 198.51.100.0/24

Security Engineering - Landscape and Policies

Introduction

- What is security by design (default)?
    - Design the system with security in mind from the beginning
- What is privacy by design (default)?
    - Consider the privacy of all data that you collect when you design the system
- What are software requirements?
    - Specifications of what the software is supposed to do
- What are the policies that you must comply with at the University of Southampton?
    - For example, you must not give your password to anyone else

- In security engineering we follow these steps
    1. Select a framework
    2. Select a maturity model
    3. Develop a security architecture
    4. Prioritise security investment and effort

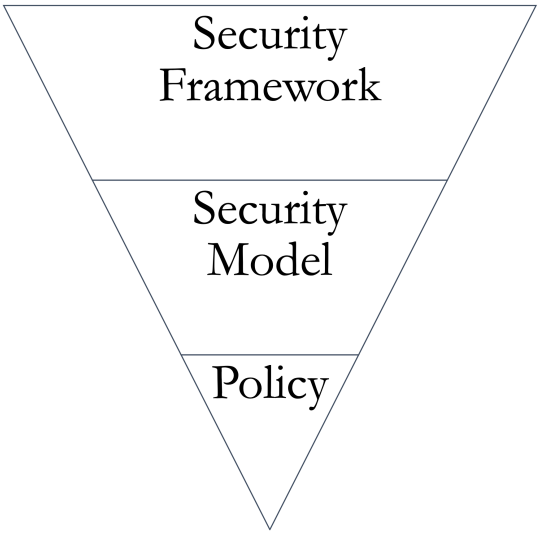
Framework, Model & Policy

- What is a framework?
    - A framework provides tools to implement/build a system
    - Security frameworks are ISO 27001, NIST, BSI

- What is a model?
    - A Model is a description of the system you want to implement/build
    - Models utilise some features that are offered by some framework
    - Examples are IoT (Internet of Things), Blockchain, Cloud

- What is a policy?
    - Policies are created based on a model
    - It is a statement that reflects the rules that everyone should abide by
    - Good policies include a purpose, policy compliance, date tested, date updated and a contact

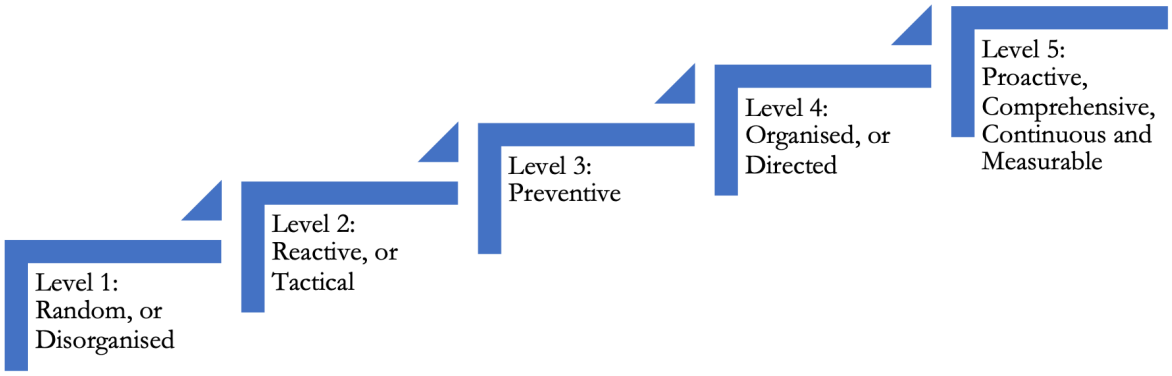
- A triangle upside down



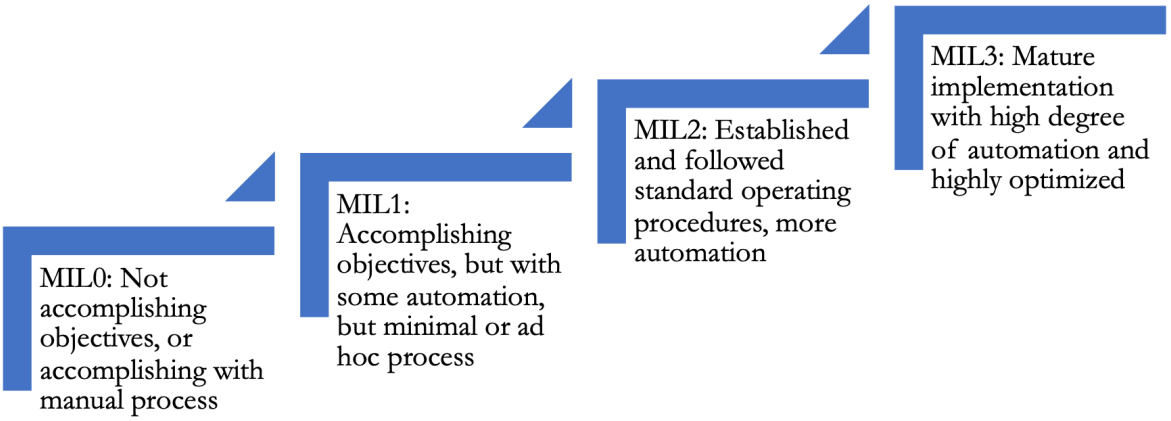
- We first select a security framework, then develop a security model and finally define the policies - this is what the triangle shows

Maturity Model

- What is a maturity model?
    - It is used to measure how good the security of your system is
- There are different maturity models
    - Here is one for networking, called the SANS security maturity model



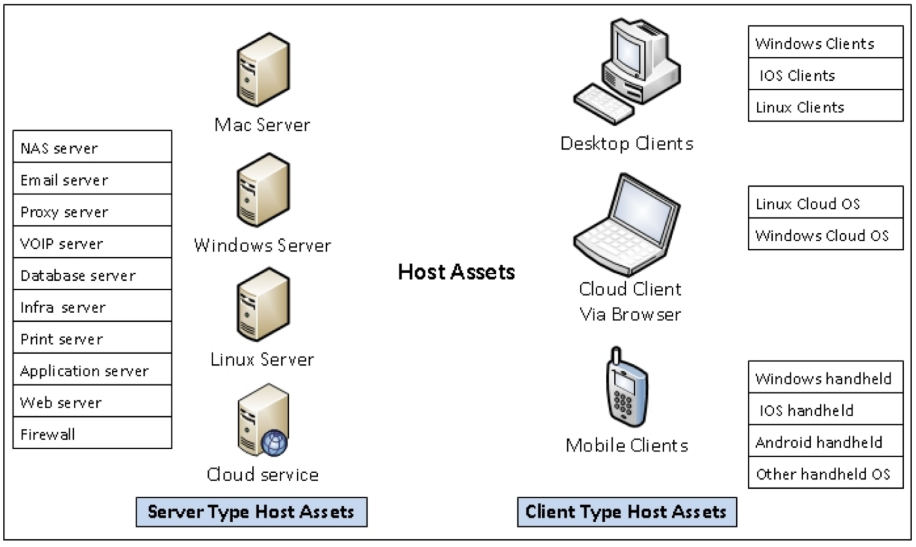
    - Here is one for code, called the cybersecurity capability maturity model (C2M2) that shows different maturity indicator levels (MILs)



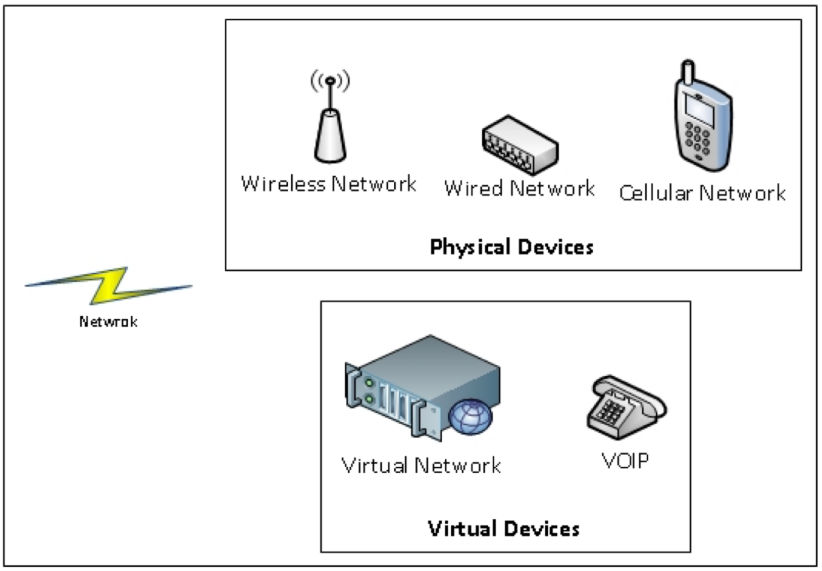
Security Architecture - Reference Diagrams

- We use reference diagrams to visualise the security architecture
- We need to
  - define hosts
  - define network(s)
  - define stakeholders
- Why do we use a diagram?
  - We can reuse (parts of) it
  - We can illustrate threats
  - We can show the impact on assets
  - We can show the impact on stakeholders
  - We can show them to managers

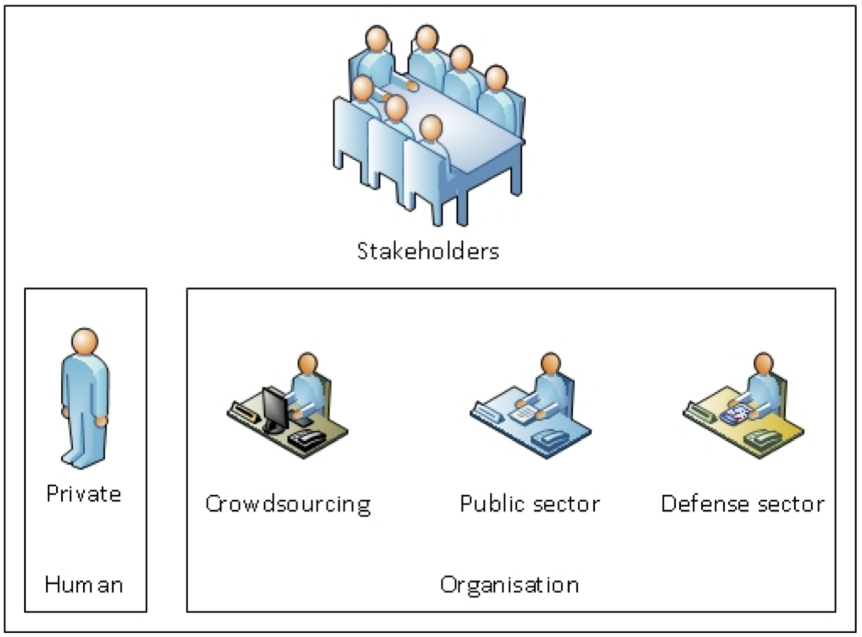
- What are hosts (or hosts assets)?
  - Host assets are typically equipment that has an operating system and does not typically include firmware
  - Here is an example showing different kinds of hosts



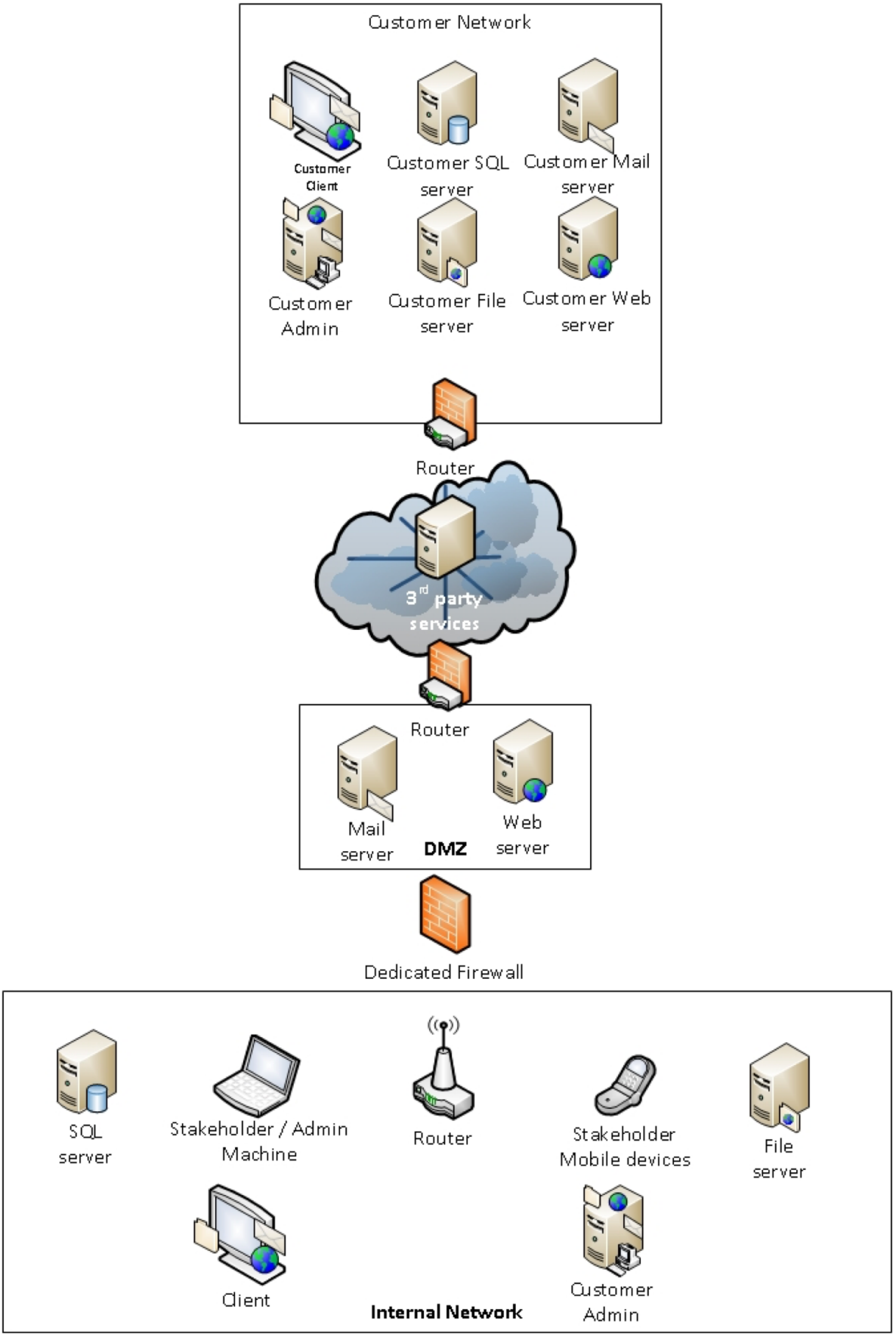
- What are network assets?
    - Network assets are used to enable communication
    - Here is an example showing different kinds of network assets



- What are stakeholders?
    - Stakeholders are people involved in the day to day operation of the organisation
    - Here is an example showing different kinds of stakeholders

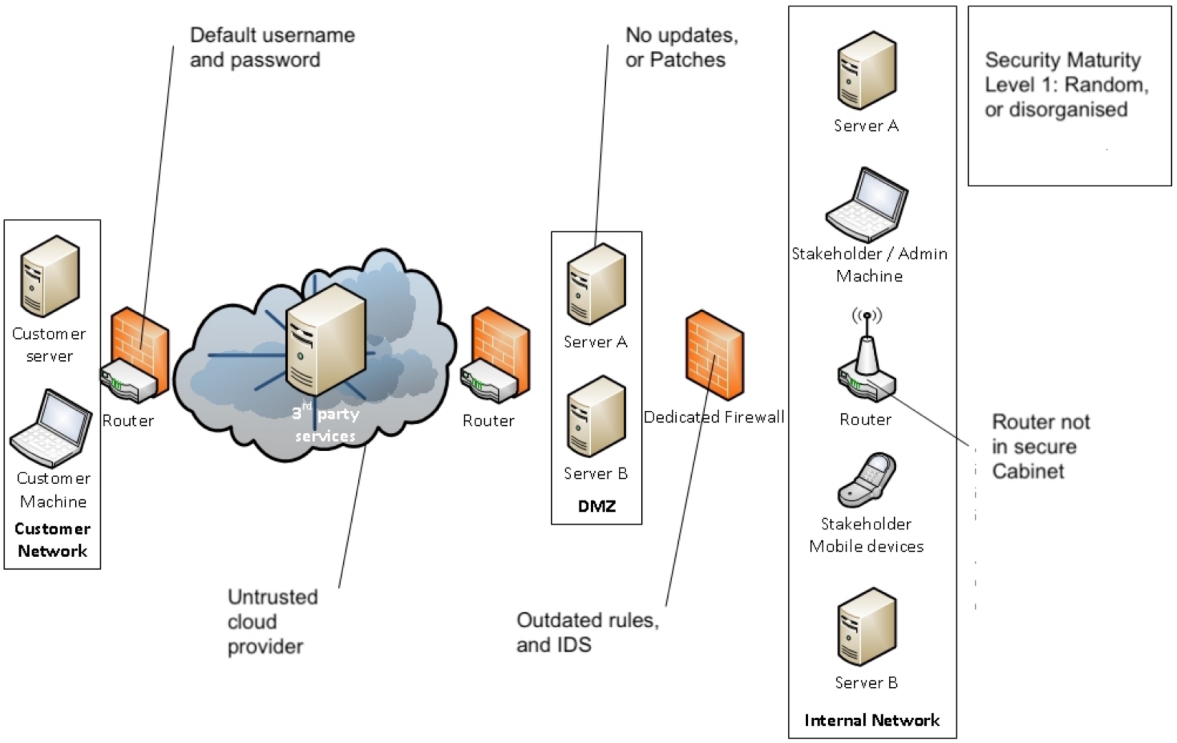


- Now that we have seen all the different bits, here is a full reference diagram


Customer Network
Customer Client
Customer SQL server
Customer Mail server
Customer Admin
Customer File server
Customer Web server
Router
3rd party services
Router
Mail server
DMZ
Web server
Dedicated Firewall
SQL server
Stakeholder / Admin Machine
Router
Stakeholder Mobile devices
File server
Client
Internal Network
Customer Admin

Threat Analysis

- Once we have a reference diagram we can use to to carry out threat analysis
- Here is an example reference diagram with threat analysis



Policy Planning

- We can write policies for
  - People
  - Network
  - Software

- Policy planning for people is education about the best security practices

- Policies for networks should be
  - Proactive
  - Comprehensive
  - Continuous
  - Measurable
- However, in industry it is often
  - Random
  - Disorganised
  - Reactive (after an attack has happened)

- The main rule for distinguishing between users and network policies is asking yourself the question: Is this policy measureable/checkable (by some system)?
    - If it is measurable then it is a network policy otherwise it is user policy
    - For example: The password must be at least 8 characters long
    - This is a network policy because it can easily be measured/checked by a system

- This is it, you have just reached the end of the cybersecurity notes
- Good luck for the exam!