

COMP6441 Student Collaborative Course Notes

W1 Lecture 1.....	3
Think of how someone could break into the house?	3
Reflections.....	4
Exercise – Foyer Security Analysis	4
Reflections.....	5
Mindset.....	5
Four Phases in the Development of a Security Mindset: D-A-E-H	5
W1 Lecture 2.....	6
6841 Lecture 1 - Buffer Overflow.....	8
How a buffer overflow works	9
Harvard vs Von Neumann architecture	9
How to play the challenges	16
GDB.....	16
W2 Lecture 1.....	17
Security by design	17
The Silver Bridge disaster.....	17
Low Likelihood and High Impact Risks	18
W2 Lecture 2.....	21
6841 Lecture 2 - SQL Injection.....	25
W3 Lecture 1.....	26
W3 Lecture 2.....	29
6841 Lecture 3	32
W4 Lecture 1.....	35
W4 Movie: The Truman Show.....	41
6841 Lecture 4 – JavaScript Injection (XSS)	42
What is JavaScript?	43

What is XSS?	43
Notable examples of XSS Exploits	43
Types of XSS:	43
Stored (P1)	43
Reflected (P1 / P2).....	44
Self	44
DOM (Document Object Model) (NOT ASSESSED)	45
Example code	45
Challenges this week: Stealing cookies.....	46
W5 – Tuesday.....	54
Authentication (short context for law people)	55
Privacy	56
6841 Lecture 5 – Format String Vulnerabilities	57
W7 – Monday	61
Integrity.....	61
The Luhn Algorithm	61
Credit Card Security.....	61
Checksums	62
Types of Hashes.....	62
Attacks on hashes.....	63
Surveillance podcast.....	64
Week 7 Tuesday Notes:	66
Week 7 -Thursday: Pentesting	70
Recon	71
Week 8 – Monday	72
Week 8 – Tuesday: Errors/Safety/Security.....	75
Week 9 – Monday	79
Week 9 –Tuesday.....	81
Week 10 Monday.....	82

W1 Lecture 1

- Don't touch people while doing the course (????????)
- Don't touch UNSW systems
- **Don't be a dick**
- Get consent before hacking someone (ethical hacking only)
- This is a security engineering course
- Someone got a corvette as a signing bonus at an American cyber security company
- Back in Richard's early days, he kept his mother's cooking recipes on his computer
- Wargames was a movie made when hacking was almost pointless
- The stakes for computer security are very high today, since most of our lives are controlled through computers. This shift towards computers controlling things was fueled by computers being cheaper than human labour. This makes defence against hackers a very high priority.
- Sit a different seat at each lecture
- People with bad intentions will analyse your habits and take advantage of them. Don't be consistent.
- The easiest way to break into a system is through social engineering, for example trading a pen for a colleague's password -> this is real btw.

Monday - 6-8 Mandatory for comp students, optional for law

Tues - 4-6 Mandatory for everyone

Tues - 6-8 Movie Night

Law seminar Tuesday 2-3:30 (Quadrangle G031) mandatory for Law students, optional for COMP

Extended programming seminar Thursday 6-8 (Ainsworth G03)

Conference on week 4

Think of how someone could break into the house?

- **Exploiting access points or weaknesses:**
 - Look for a hidden spare key, walk through an open door, pick a lock, or break a window or door
 - Wait for them to leave the door unlocked

- House keys inside the car
- Dig a tunnel under their house
- Airstrike
- Brute force garage door (brute force controls)
- **Using social engineering or deception:**
 - Pretend to be an electrician, government official, police, church member, delivery man, or salesman
 - Play the long game and marry them or pretend to be their friend
 - Bribe child to open the door
 - Coercion/blackmail
- **Leveraging timing or situational opportunities:**
 - Wait for them to go on vacation
 - Hide in the car and drive into the garage
 - Wait for them to leave the door unlocked
 - Make them believe there's danger and flee the house
- **Directly obtaining or duplicating access tools:**
 - Take key from person
 - Lock impressioning
 - Clone pass card

Reflections

- So many ways to break into a house yet when we lock our houses we think they're safe
- **Preconceptions are death** in security → Attackers exploit trust, assumptions and overlooked vulnerabilities
 - "Safe as Houses" – Common yet often misguided assumption that something is secure → can lead to overlooked vulnerabilities
- Physical security is foundational → software security is pointless if an attacker can easily gain physical access
 - Doesn't have to be tamper-proof, sometimes it's sufficient to be tamper-evident
 - E.g. Fire extinguisher, sealed lid on jar, bottle cap

Exercise – Foyer Security Analysis

- Observation exercise in assessing the security features of a foyer.
- Initially, most participants rated the foyer highly secure (4-5 out of 5).
- However, when tasked with breaking in using only found items, each cohort discovered vulnerabilities within minutes.

Reflections

- How is it that everyone thought it was very secure yet was easily able to be broken into in a few minutes without even using any special tools?
 - They were only looking at the strengths
 - The way the question was asked was priming them to only look at the strengths
 - Defenders often focus only on strengths, while attackers seek out weaknesses
→ **Attacker and Defender Mindset**
- A cybersecurity professional must adopt an “attacker mindset” to recognise vulnerabilities effectively

Mindset

- **Asymmetry** in defence and attack:
 - Attacks only need to exploit a single weak point, while defenders must secure all points
- Biggest danger to a security person:
 - Someone is thinking they’ve solved the problem
 - **Overconfidence** (hubris)
- Developing **"Security Eyes" / Security Mindset**:
 - Learning to view every component and interaction as a potential security risk, whether in software, physical setups, or user behaviour.

Four Phases in the Development of a Security Mindset: D-A-E-H

- **D – Defender mindset (“Barbarian”)**
 - Basic defensive mindset, securing only against known threats and strengthening current defences
- **A – Attacker mindset**
 - Learning to think like an attacker, probing for weaknesses, using creativity and an understanding of human nature to exploit flaws
- **E – Engineering Mindset**
 - Learn from professionals with an engineering mindset and practice to improve quality of security measures
 - Measurement, estimation and calculation
 - Skepticism

- Testing
- Review
- Openness/transparency – never just “trust us” (top men)
- Treatment of errors
- Standards
- Professionalism
- Closing the loop
- **H – Humans**
 - We must understand that in our system, humans are everywhere.
 - Systems should be designed to account for the weakness of humans.
 - It’s perfectly good to make a mathematical system which is super complex, but a real secure system must account for the fact that humans aren’t perfect logical beings.

W1 Lecture 2

Top. Men. (don’t worry = start to worry)

Security theatre = only looks secure (but maybe is not)

- cybersecurity measures that only offer a semblance of security without delivering substantial protection
- Examples:
 - Document shredder
 - Bike locks
 - Airport security

Thankyou <3

Lecture stream link: <https://www.youtube.com/live/xpdGI-RDXnA>

bugs->subset of bugs->vulnerable to attacker (zero day vulnerability)->exploitation rip

- Defence mindset, Attacker mindset, Exploitation
 - **Note from Daniel Chen** - I think E is actually Engineering, upon rewatching the lecture. Not sure where you got Exploitation from

ENGINEERING! (What is it that engineers do? What do they do that makes things reliable?)

- Standardisation
 - Using the right nuts
 - E.g. IEEE standards
- Estimation & calculation
- Precision
- Experiment & testing
- Model & prototype
- Setting reasonable tradeoffs (bolster weak point first)
- Peer review
- Identify risks and workout strategies
- Constant monitoring & scalability
- Openness/transparency
- Treatment of errors
- Professionalism & duty
- Closing the loop

Why do things fail?

- Poor planning
- Human error?

Halifax Explosion

- Two cargo ship crash into each other
- Fuel spill over deck and catch a spark causing massive explosion
- Some 2,000 people died and thousands more were injured
- Towns were obliterated, a tsunami was created due to the explosion too

Recommendation and investigation to Halifax Explosion (Return of the Halifax) - need to make sure recommendations are something that can be implemented by the one you're recommending to

1. A Wider canal for flexibility
2. Translator/standardised communication method with lighting or horn (e.g. morse code)
3. Better cargo protection
4. Licensing for entering and exiting the canal

5. Monitoring behaviour on the canal (guard posts)
6. Better brake system & defensive cruising mindset
7. Limit dangerous goods a ship can carry
8. Evacuation procedure (Baguette Baguette Baguette)
9. Customs - be aware of what ships are coming in and what they're carrying

Recon

- Start of attack
- Active & passive
- Find the easiest-to-access vulnerabilities

Something Awesome, around 30hr work

- Audit something
- Make something
- Learning something
- Teach something
- 6841 - Must be technical

6841 Lecture 1 – Buffer Overflow

<https://phrack.org/issues/49/14.html>

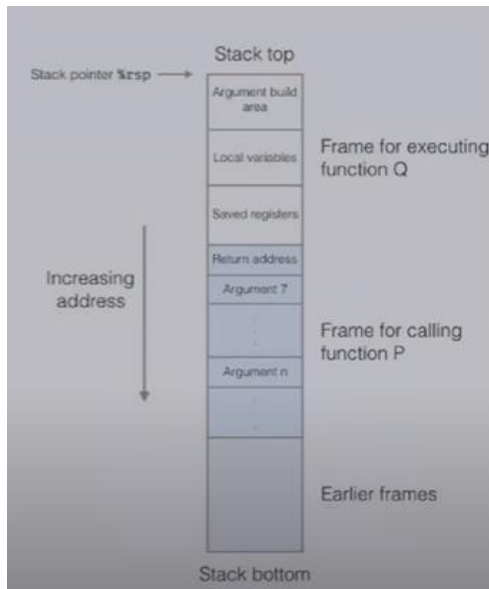
Buffer overflows exploits IRL:

- WannaCry
- Heartbleed
- Adobe Flash vulnerability
- Fortinet vulnerability

Don't touch uni infrastructure (Full stop. This includes webcms and moodle)

How a buffer overflow works

A buffer of data (e.g. characters) sits in memory



THE STACK GROWS UP (like a stack of pancakes!)

Stack stores things in a logical order, last in = first out - used for storing local variables, stack frames etc.

Stack frames are virtual address spaces to store a function's allotted memory, both data and control information (problem)

This means one could inject arbitrary data into the portion "meant for control" -> bad stuff happens

Harvard vs Von Neumann architecture

Harvard	Von Neumann
Physically separate piece of memory to store control variable	Same memory stack for both data and control
Two physical memory types	Two memory items in one physical place
Was more difficult to implement but nowadays we partially shift to it again (e.g. shadow stack)	Simplicity and efficiency

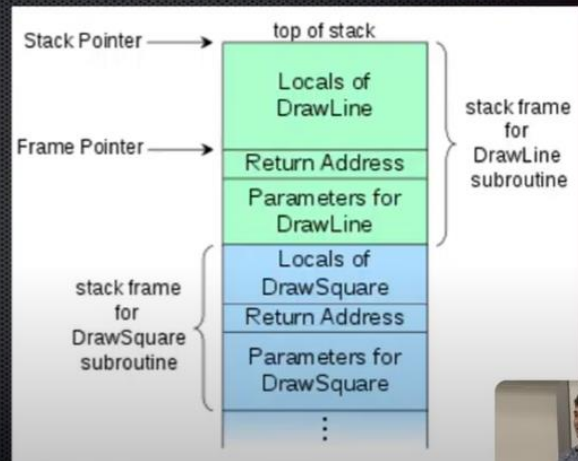
Von Neumann architecture is fast and elegant, but mixing data and control (risky)

Overflow into control structures ☹️

Nowadays we partially shifted (i.e. Frankensteined) back to Harvard architecture to avoid buffer overflows

A Stack Frame on x86 Architecture

- In this example, DrawSquare() calls DrawLine() four times.
- The stack frame of a called function includes:
 - The parameters passed to the function
 - The return address to go to when the function returns
 - The base pointer - the bottom of the stack of the calling function



Function Call Process

From the perspective of the calling function

Definition: `int sum(int x, int y) => sum()` takes two integers and returns the sum.

1. "I want to call `sum()`"
2. "First I need to give `sum` the parameters to the function" => push to the stack the values of `x` and `y`
3. "I need to return to this point in code execution when `sum()` has completed" => push the return address of `sum` to the stack
4. "I need to remember where my own stack stops when `sum()` returns" => push the base pointer to the stack
5. Start execution of `sum()`

Function call process:

1. Call `sum()`
2. Push parameters (values of `x` and `y`) to the stack
3. Push return address of `sum` to the stack
4. Push base pointer (end of parent stack) to the stack
5. Execute `sum()`

Even if you call a function multiple times (e.g. using 4 lines to draw a square), only one stack frame at a time - only one function is actually called at once

Note from Daniel Chen regarding function calling in x86:

I don't think the lecture slides are quite right here, in x86 (32 bit) it should generally be:

1. Caller pushes arguments **in reverse order**
2. Caller uses **call**, which jumps to the callee as well as pushing the return address
3. Callee pushes EBP of caller to stack (and sets EBP register for callee but this isn't on the stack), alternatively **enter** does this but isn't usually used
4. Callee executes body
5. Callee uses **leave**, which "pops" all locals (not really but you can treat it like a pop), and pops / restores EBP off the stack
6. Callee uses **ret**, which pops the return address of the stack and jumps to it (at this point everything except for arguments is now popped).
7. Caller pops arguments off the stack.

That is, from the "perspective of the calling function" (what the slides claim to describe), it does **not** push the base pointer (this is the responsibility of the callee), and steps 3 & 5 are really the same step (**call**).

Additionally, one extra step should be added at the end, being that after the function returns, the caller is responsible for popping arguments off the stack (it pushed, so it should also be in charge of popping, i.e. the **call**'s stack usage is transparent to the caller).

Also worth noting is that the base pointer is not actually the "end of" a function's stack. It points sort of in the middle. The address being stored (i.e. 0 offset) is actually the stored EBP of the parent that we pushed earlier, and toward the higher addresses ("lower down" on the stack, positive offset from EBP) is the return address and arguments, whereas towards the lower addresses ("higher up" on the stack, negative offset from EBP) lies locals.

You can experiment with / verify this here: <https://godbolt.org/z/dYfssj8xG>

Demo 1 (All Hail the demo Gods)

```
1 #include <stdlib.h>
2 #include <stdio.h>
3
4 int main(int argv, char **argc) {
5     char name[16];
6
7     printf("Welcome!\n");
8     printf("What is your name?\n");
9
10    printf("> ");
11    gets(&name);
12
13    printf("Welcome, %s!\n", name);
14    return EXIT_SUCCESS;
15 }
16
```

Q: What happens if you put more than 16 characters in "name"?

A: segmentation fault! "gets" reads from stdin into the buffer beginning at the address of "name"
-> can go past the allotted space for "name", past the end of the stack and into the return
address of "main"... a **buffer overflow**!

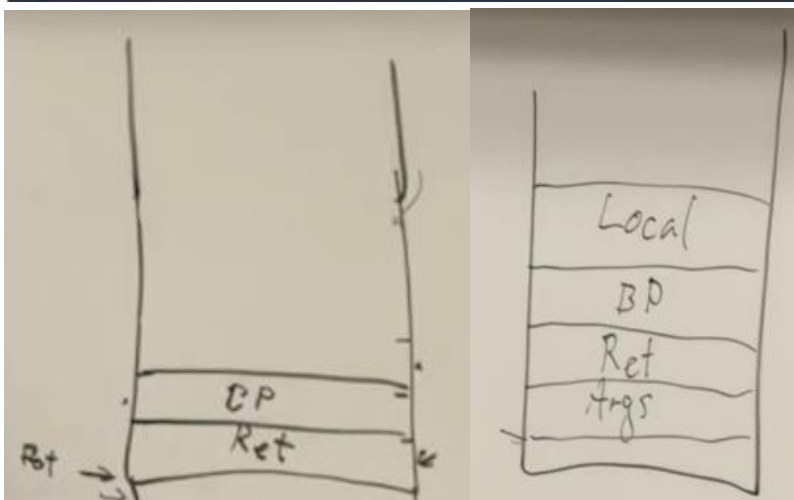
Never use "gets()"; it can't check for buffer overflow.

Demo 2

```

1 #include <stdlib.h>
2 #include <stdio.h>
3
4 int printName();
5
6 int main(int argv, char **argc) {
7
8     printf("Welcome!\n");
9     printName();
10    printf("Everything is good.\n");
11    return EXIT_SUCCESS;
12 }
13
14 int printName() {
15     char name[16];
16     printf("What is your name?\n");
17     printf("> ");
18     gets(&name);
19     printf("Welcome, %s!\n", name);
20     return EXIT_SUCCESS;
21 }
22

```



Main has no local variables, so the stack reserves space for the return address and then the base pointer. (Base pointer holds the address for the bottom of the stack for the function that you are in). It then stores space for the local variables in printName(): 16 for the array name[16], and then the return address, and base pointer.

Q: What happens if you put more than 16 characters in "name"?

A: return address and possibly also base pointer are overwritten -> the computer will jump to whatever bytes are now specified -> probably segmentation fault because the code can't be executed.

Or you might just overwrite some local arguments that aren't used, so there is no real consequence

Or you could overflow the return address with the exact same value as before...

Anything could happen, fun!!

What can you do with this?

Make the return address some other legitimate address that isn't dereferenced -> arbitrary code execution

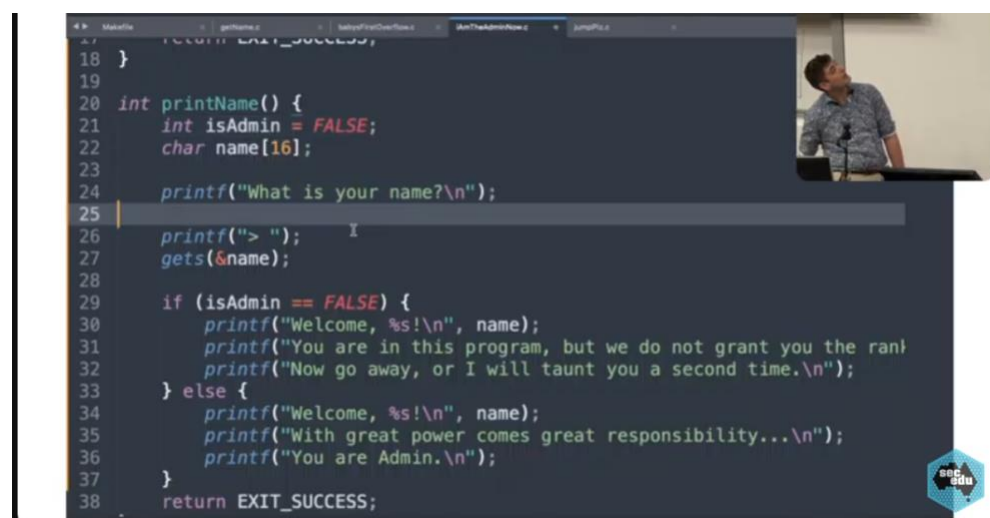
"fgets" is safer - why?

You have to pass in the buffer size too (but if it's wrong then you get buffer overflow anyway)

What happens if you only overflow the base pointer?

Probably nothing immediately, but then the calling function doesn't know where to stop anymore -> just keeps on popping and madness happens

Demo 3 (isAdmin)



```
18 }
19
20 int printName() {
21     int isAdmin = FALSE;
22     char name[16];
23
24     printf("What is your name?\n");
25
26     printf("> ");
27     gets(&name);
28
29     if (isAdmin == FALSE) {
30         printf("Welcome, %s!\n", name);
31         printf("You are in this program, but we do not grant you the rank\n");
32         printf("Now go away, or I will taunt you a second time.\n");
33     } else {
34         printf("Welcome, %s!\n", name);
35         printf("With great power comes great responsibility...\n");
36         printf("You are Admin.\n");
37     }
38     return EXIT_SUCCESS;
}
```

Don't run random pieces of code (at least use protection e.g. virtual machine or a second device)

Name = "AAAAAA..." -> overflows into local variables, setting isAdmin to something other than false -> now you have admin! (and then it breaks because you also overflowed the base pointer and potentially the return address)

What happens if you declare "name" first? - the same thing, due to compiler optimisation - local variables are always stored in the same order *no matter how you declare them*.

To successfully just change to admin without causing any further issues, you can simply type a name of 20 characters, which overflows exactly into isAdmin (which is a 4 byte integer) and overwrites it. Since the program only checks if isAdmin is not false, it will return true for any character.

A simple solution - explicitly check that isAdmin is true (1) if it is not set to false (0)

Demo 4 (gdb)

Gdb displays addresses and what they point to??

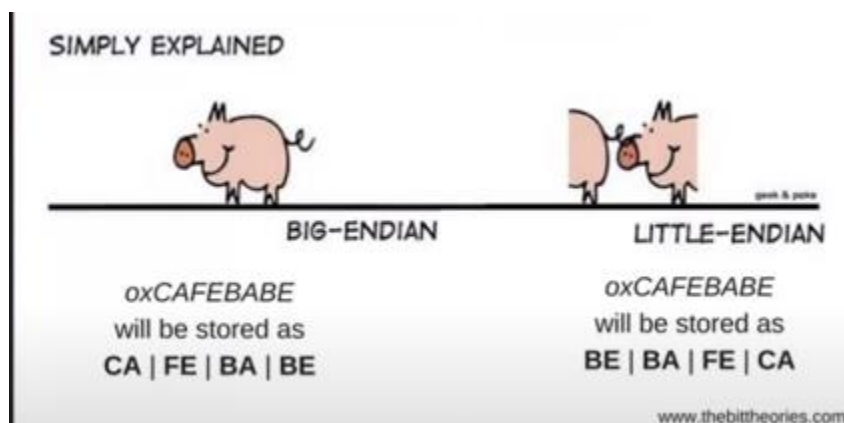
What if we store the admin permissions in a separate function?

We could overflow the return address so that the first called function returns not to main, but to welcomeAdmin() which is actually never called.

NOTE: Due to addresses being stored in Little Endian, you have to write the address backwards. For example, if the address is 08049268 (hexadecimal), you should be using `b'\x68\x92\x04\x08'`.

(for 32 bit we need 4 bytes, the bytes are reversed due to little endian as above)

Endianness



Bytes are reversed, bits are not! The bits are always stored "in order" - smaller address to larger address

In little endian, the least significant byte is stored at the start (reverse order).

In big endian, the least significant byte is stored at the end (normal order).

"Endianness is the worst decision ever" - Kristian Mansfield

How to play the challenges

You get a binary without the flag, Kris will host a binary with the flag -> connect remotely with netcat/ncat and run the correct exploit on Kris's version to get the flag

A "payload" is another program that runs the exploit

GDB

https://cgi.cse.unsw.edu.au/~learn/debugging/modules/all_gdb/

Run the program with GDB - disass (disassemble), set breakpoints, see what's in memory...

Pay attention to EBP (base pointer), ESP (stack pointer, top of stack), EIP (instruction pointer, next instruction to be executed)

"pwndbg" makes GDB much more palatable

Protective mechanism

- Stack canaries - extra local variable which will be hit by the buffer overflow before anything more important, it dies for the hacker's sins
- Position independent executable - jumps refer to "relative to this address" rather than absolute memory positions - much more work for a hacker to figure out where their exploit is relative to the instruction pointer
- Address space layout randomisation - randomising the addresses of the stack, heap, base pointers etc.
- DON'T TAKE MORE DATA THAN NEED aka get good

"Have you tried being a better hacker?" - most of your tutors if you ask a simple question

W2 Lecture 1

- Estimate probability based on previous cases/data
- Risk is invisible, luck is involved at times
- Always have a plan B, what will happen if everything goes wrong?
- There are a million things (like clouds - by Luke Howard) that can go wrong, but we should look for patterns

Security by design

- Anti-pattern (complexity) - where insecurity grows & defender mindset
- Dual control - needs 2 (hopefully) independent things to happen first e.g. turn 2 keys to launch the missile - hacker tries to make the 2 things less independent
- Defence in depth "if something goes wrong, it's not over" - don't only rely on one thing (always have a backup)core
- Weakest link: strength of a security setup is the weakest point in the setup
 - Example: nuclear launch, with the president being a single point of failure. But not really because there are humans in loop.
- Single point of failure - if it fails the whole thing goes to pieces - you don't want this
- Have failure points be visible so it is obvious when these are compromised

How to defend against the defender mindset?

Attacker mindset

Look for the "weakest link" - that probably also means examine everything

Have a dedicated "red team"/attack team - pentest each other

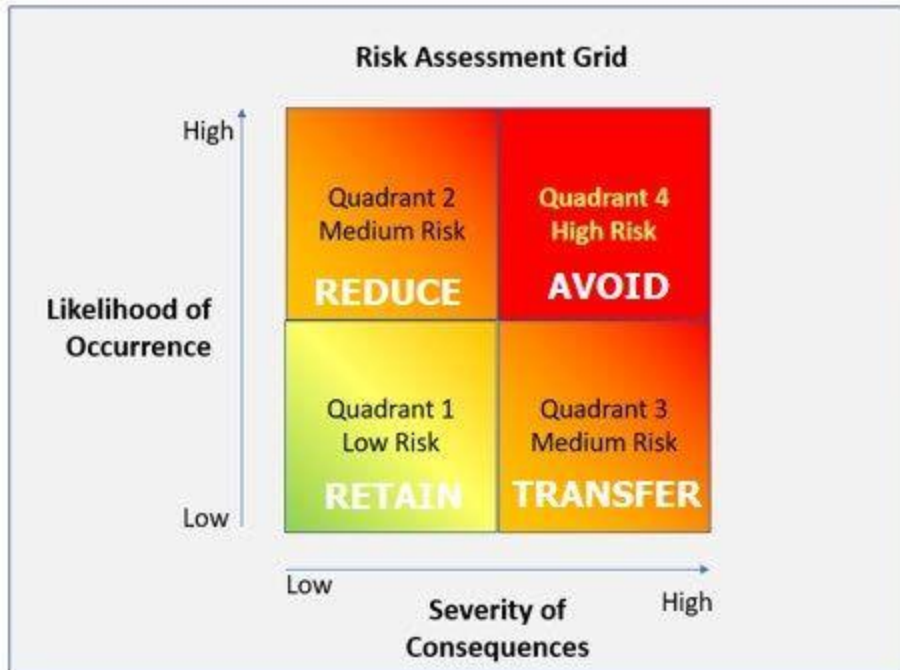
But what if the attack team gets compromised/conflict of interest (because usually they're already affiliated with you)? Hire a consultancy firm?

The Silver Bridge disaster

There was a crack because of a fault in design allowing water to pool and corrosion to occur. No supports outside the one point meant the entire bridge went down when one joint broke (single point failure)

Risks

Quadrant diagram - likelihood + impact



Published by NIST (National Institute of Standards and Technology - USA).

There can be 4 types of risks:


1. Low likelihood and low impact - we can mostly ignore these risks
2. Low likelihood and high impact - very difficult to predict and assess
3. High likelihood and low impact - easier to make predictions about due to history
4. High likelihood and high impact - very dangerous

Low Likelihood and High Impact Risks

- How well prepared were people?
- What was done in advance to deal with the risk?
- What should have been done?
- What were the warning signs?
- Did people ignore the warning signs?
- What lessons were learnt afterwards?
- Desire to blame vs desire to improve?

The full list of examples from Richard's slides can be found [here](#)

Low likelihood high impact risk examples

- Tsunamis hitting the coast of Sydney
- Microsoft goes down -> can't sign in or authenticate (e.g. crowdstrike)
- Uni network going down (this happened last week )
- Earthquakes
- Prions (misfolded protein that's infectious)
- Asteroid strikes
- Having a stroke or heart attack happening right now (its more common if its over your lifetime)
- Bit flip
- Bank run (where all the customers withdraw their accounts, mostly because they believe that the bank might fail)
- Terrorist events (little expenditure, maximum human impact)
- COVID or similar pandemics (to be honest with around 8 billion world population we should expect pandemics at this point)

Creeping determinism - "Monday Experts", what they should have done

After something happened, you view it as inevitable and find out what you could've done to prevent it.

You don't want solutions to come after the events have already happened, try to think about these things at least once and assign priorities accordingly.

When a risk is prolonged, people tend to ignore it

Problem with risk: the humans, us

10. Organisation or team/groups (people are flawed)
11. You, yourself (overconfident, biases and normalised deviance)

Pseudo-certainty effect — the tendency to make risk-averse choices if the expected outcome is positive, but make risk-seeking choices to avoid negative outcomes.

Try not to leave the logbook summaries to the last minute

- Can use docs to copy paste it across into the page editor for maximum efficiency

in advance

- How well prepared were people?
- What was done in advance to deal with the risk?
- What should have been done (with the benefit of hindsight)?

in the immediate leadup to the incident where the bad risk outcome eventuated

- what were the warning signs? did people ignore warning signs?

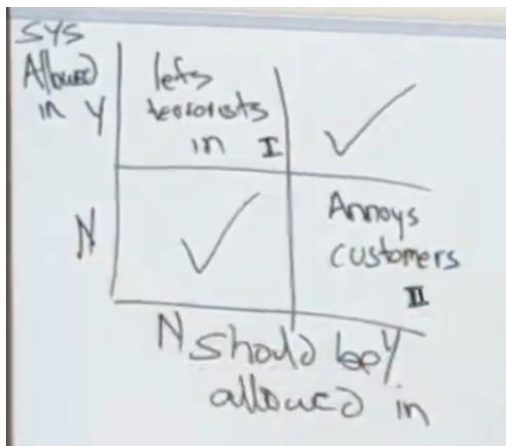
during

- how was it dealt with while it was happening? (incident response) - strengths and weaknesses

Afterwards

- What were the lessons learned?
- Do people seem to be learning them?
- notice the desire for blame vs the desire to improve in the aftermath

There are two ways a system can fail - you can go in but you shouldn't, or you can't go in but you should



Type I (false positive)

Type II (false negative)

The system shouldn't allow you in but it does	The system should allow you in but it doesn't
Convicting an innocent person	Acquitting a guilty person.
A medical test diagnosing a healthy person with a disease	A medical test failing to diagnose a sick person

NB when referring to Type I/ Type II errors you can refer to these as “Type I and Type II” errors since they are attributed depending on how you phrase the scenario (e.g. security letting people in vs security preventing imposters)

Y = yes, N = no, left side is “system allows you in”, bottom is “should you be allowed in”

There’s a tradeoff - it’s easy to reduce one error while increasing the other. Reducing both is difficult.

Quite often people cannot see both false positives or negatives at the same time e.g. when it comes to indicting the innocent or letting the guilty go free.

Usually, you want a system to fail in a way that is visible e.g. in the airport security example it is harder to tell false positives than false negatives since annoying people is easier to see than imposters sneaking onto planes 🧑🏻‍🚒 .

Find the other error that was not intended by the developers:

W2 Lecture 2

[Richard's Slides](#)

Mitigation - either reduce the likelihood or impact

Pass on the risk - insurance

Immunisation - hedging

Accept the risk

Everybody pool the risk together

- Contracts are commonly used to mitigate risks to a business e.g. Disney Allergy situation

Anatomy of a typical attack

1. Patch: This refers to vulnerabilities in software or hardware that can be exploited by attackers / social : manipulating people to gain unauthorized access or information.
2. Privilege escalation: gain control over more critical systems or data
3. Persistence
4. Stealth
5. Life expectancy of a new machine connected to the internet in minutes (sans)

History of Cyber Security

Earliest hackers were individual people.

In the earlier days, hackers used telecommunication networks. They were called phone phreakers.

The phones used the audio signal to send the control data. The control signal for getting operator mode was 2600 Hz which was coincidentally the frequency of a whistle given by the popular cereal "Captain Crunch".

When Microsoft Money came around, hacking took a different turn as hackers realised that they could get money from hacking.

Hacking then became a coordinated group activity.

- The value of cybercrime exceeds that of all regular crime combined (it's why cybersecurity roles are so high in demand)

Pattern - mixing data and control

- [video clip](#): the data given dictates what happened next

A dangerous yet common scenario in software design where data (from users, content, etc) are not separated from control mechanisms (commands, logic, instruction). This means that data can be sent as control mechanisms, leading to huge security vulnerabilities.

Nature of secrets

- security is the absence of insecurity → there is no such thing as security there is only insecurity
- there is no such thing as a secret there is just the absence of knowledge
- How to keep a secret? → don't tell anyone
- The optimal number of people to share a secret and ensure it stays a secret → 0
- Can't trust people to keep secrets
- If you tell someone the secret, it is hard for you to control them to not tell the secret to other people
- How do you know if someone knows a secret?

Can you design a retractable secret?

- attack the person that knows the secret
 - undermine their credibility
 - kill the person
 - medically induce amnesia
- disguise the secret
- give them a secret you can repudiate
- ultimately you can't really retract a secret once it's out
- spread of information is monotonic
- Give fragments of the secret
- Give an indirect access to the secret and not the secret itself

Cryptography is the art of writing and solving codes.

Cryptographic properties:

C - Confidentiality

I - Integrity

A - Authentication

Steganography is when you have a message but you conceal the existence of the message itself.

Ex: tattoo the message on the bald head of a slave and let the hair grow

Design principles for military cyphers

1. Practical system, not mathematical/indecipherable
2. Can be given to the enemy without inconvenience, not required to be secret
3. Communicable and retainable keys, also changeable/modifiable at will
4. Applicable to telegraphic correspondence
5. Portable documents and usage must not require a collective of people
6. Easy to use system, mentally and intellectually

The best way to keep a secret is by making it not seem like a secret!

Security by Obscurity: Improve security by hiding the details/mechanism of the system, but then it makes the system close-sourced and vulnerable to invisible bugs

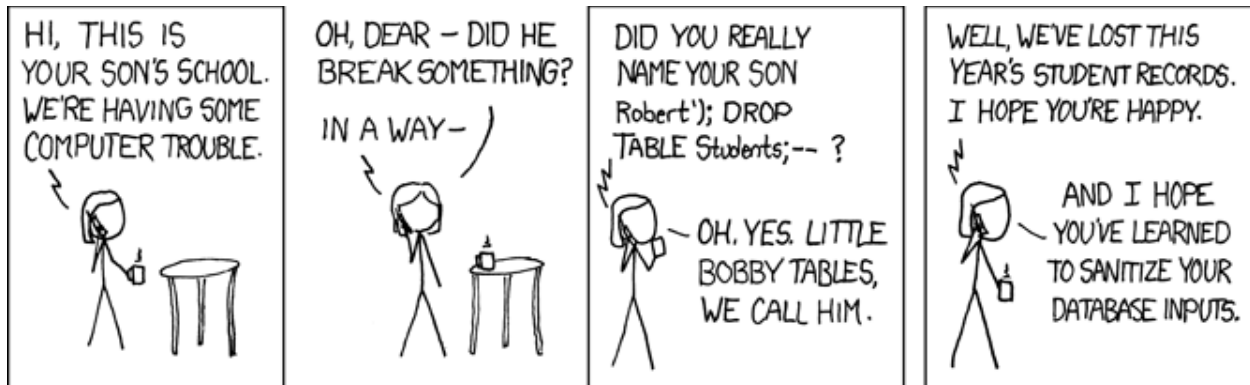
Cryptography Properties

- Authentication: One of the core properties of cryptography. It ensures that the sender of a message is who they claim to be.
 - Authentication is the hardest property of cryptography to achieve because it requires not just the encryption of data but also validation of the identity, which involves multiple layers of security.

History of Codes

- Steganography:
 - Involves hiding the existence of a message rather than just encrypting it.
 - Useful for communicating confidential information without raising suspicion.
 - Works because an interceptor does not know the communication exists.
- Caesar Cipher:
 - One of the earliest known encryption techniques.
 - Involves taking symbols (letters) in a language and replacing them with another set of symbols (shifting them by a certain number of positions in the alphabet).
- Codes in Conflict:
 - Historically, during times of conflict, countries believed their codes were uncrackable.
 - This reflects a defender mindset, where nations overestimated the security of their cryptographic systems and underestimated the ability of adversaries to break their codes.

6841 Lecture 2 – SQL Injection



Database:

- Structure and organisation of information
- Persistence across restarts
- Efficiently storing big data

Use a UML diagram to map table attributes and relationships in a structured database

Don't store passwords - use a hash or ask another acting directory?? (single sign on)

There's always something that could go wrong

Database engines are not designed to tell the difference between code and data. SQL Injection is when an attacker can modify the data used in an SQL query such that it is executed as code.

- Attacks aim to manipulate the input string into commands

E.g. `test' OR '1'='1';--` → `SELECT name FROM students WHERE name = 'test' OR '1'='1';--` AND `password = 'password';`

```
SELECT * FROM users WHERE username = 'test' and password = 'test';
```

Imagine this is how a system works. To gain access, we can add an always true statement into the username and password. Note that the " between the names are hardcoded. This is how we can hack into it.

```
SELECT * FROM users WHERE username = 'test' OR '1' = '1' AND password = 'test' OR '1' = '1';
```

How would we type this into the login and password in an application?

Login: test' OR '1' = '1

Password: test' OR '1' = '1

We do not add the ' at the beginning or end since it is hardcoded.

W3 Lecture 1

No tutorials this week! Just help sessions

You can submit a poster of your project for the conference next week

“Hacker Hangouts in the seclab tomorrow at 1pm” - Kris – You can also join secsoc.

Information measure: quantity that calculates length of the shortest message needed to transmit information.

The concept: when we say something in words, the other person can reconstruct the information.

Language has a lot of redundancies (entropy), and we could use patterns in them to crack Cipher.

CIA Triad – Confidentiality, Integrity, Access & Availability

Confidentiality – Only sender & receiver should “understand” the contents of a message, usually achieved via encryption

Integrity – Sender/Receiver want to ensure messages between them are not altered and ideally achieve non-repudiation (irrefutable proof that sender sent the message)

Access & Availability – Services must be accessible to sender/receiver

*I think A is Authentication, not access & availability ?

Entropy is a measure of chaos - having low entropy means that we can make accurate predictions

- Active communications is sending something, and the person at the other end understanding what you mean
- Guessing game - Richard was thinking of something, and the group has 20 questions to try and guess what it is, each answer narrows the options down a bit further

- The best 'yes' / 'no' questions are those where both possible answers equate to around 50% of all possible solutions, eg. if I'm thinking of a number between 0 and 100, it's better to ask, 'is it below 50?', than 'is it below 5?'
- Try to get familiar with powers of 2, eg. $2^{**3}=8$, $2^{**8}=256$, 2^{**10} is approx 1000. Knowing that, you can scale up to larger, eg. $2^{**45} = 2^{**5} \times (2^{**10})^{**4} = 32 \times 1000^{**4} = 32 \times 1000 \times 1000 \times 1000 \times 1000 = 32 \text{ prevalence,000,000,000,000}$
- Calculating error rates - use a weighted average based on outcomes, as the outcome is the information you have after doing a test. Example of COVID test, if the test says Yes, it's accurate X% of the time, vs if the test says No, it's accurate Y% of the time. See worked example in Richard's slides.
- "You need to understand and be familiar with exponential growth"
- Growth rates - make sure you know the difference between linear, quadratic, cubic, exponential. The media loves to say 'exponential' when often it's something else.

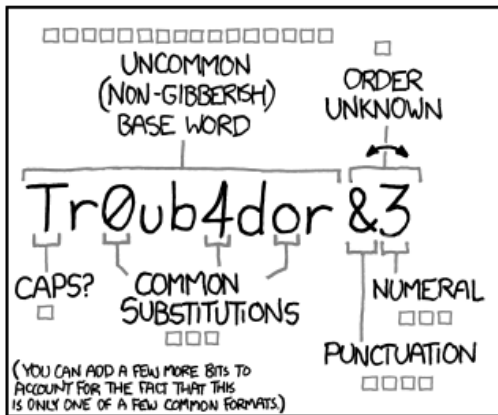
Growth => as x grows 1,2,3,4,5,.. how quickly do the results of various expressions grow?

	growth rate	x=1	x=2	x=3	x=4	x=5
4 times x	linear	4	8	12	16	20
x^2	quadratic	1	4	9	16	25
x^3	cubic	1	8	27	64	125
2^x	exponential	2	4	8	16	32

You need to understand and be familiar with exponential growth

[Wolfram Alpha](#) is a convenient way of doing calcs

- - Passwords are an example of exponential growth scaling from character length.
- Prefixes for large numbers - make sure you know the correct prefixes, eg, kilo, mega, giga, tera, peta, exa etc. There is a full list here on Wikipedia: https://en.wikipedia.org/wiki/Metric_prefix
- Computer clock speeds - say the CPU is 3 GHz, that's about 2^{31} operations per second. For a quad core computer that's $4 \times 2^{31} = 2^{33}$. Say it takes 32 steps to attempt a password, that 2^5 , so $2^{33} / 2^5 = 2^{28}$ password attempts per second = 2^{40} per hour - that's a lot! And that's what we need to consider to protect against brute force
- General rule (ignore NSA or highly sophisticated players) - 128 bits is considered enough to protect against brute force. Note that in the password example - the 128 bits is for the encryption key, not the size of the password itself



~28 BITS OF ENTROPY

$2^{28} = 3 \text{ DAYS AT } 1000 \text{ GUESSES/SEC}$

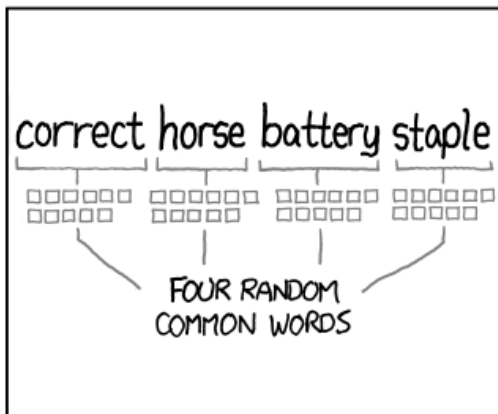
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)

DIFFICULTY TO GUESS: **EASY**

WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE 0s WAS A ZERO?

AND THERE WAS SOME SYMBOL...

DIFFICULTY TO REMEMBER: **HARD**



~44 BITS OF ENTROPY

$2^{44} = 550 \text{ YEARS AT } 1000 \text{ GUESSES/SEC}$

DIFFICULTY TO GUESS: **HARD**

THAT'S A BATTERY STAPLE.

CORRECT!

DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT

THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

W3 Lecture 2

- Help sessions all day Wednesday, Thursday, Friday
- No tutorial because of that

Interesting Tangents

- Donald Knuth: Random Number Generation
- Bruce Schneier: Great Blog

Codes vs Ciphers

Codes	Ciphers
- Require prior knowledge	- Jumble up a message using pattern
- Unless you know the code it's hard to crack	-
- With a large enough sample size, codes become a substitution cypher	-

Enigma Machine

- Substitution changes every letter
- Could not encode a letter as itself



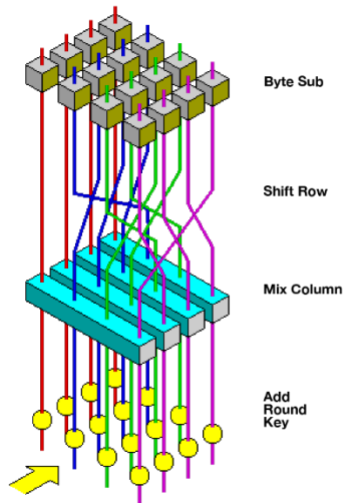
DES (Data Encryption Standard)

- Break the message into blocks (e.g. 3 bits), encrypt each block, then put the encrypted blocks together
- Iterate this encryption
- Can still be brute-forced (key is 56 bits)

- and as hardware advanced, it became easier and easier to crack
- Cracked through 'differential analysis'

AES (Advanced Encryption Standard)

- 128, 192 or 256 bits



Stream cypher:

- Encrypts as the letters are written out
- Enigma was a stream cypher

Block cypher:

- Encrypts in blocks
- Allows for easier jumbling
- AES and DES are block cyphers

ECB Mode – just concatenate all the blocks – but then the code can be broken with frequency analysis.

Side Channel Attack

- Indirect leakage of information from a related physical source
- Examples:
 - Radiation emissions from TVs
 - Spectre/meltdown
 - Pentagon pizza orders spike whenever an operation is ongoing.
 - Strava had user GPS hotspots in the middle of nowhere because of military personnel using it.
 - Seeing if lights are on late at night on a building floor of a certain corporation.

- Computer fan speed

Password Problems

- There is more security in a key, but it is more difficult to remember. This creates the risk of physically recording the key (which is a security issue).
- The whole system needs to be secure, not just component by component. Think end-to-end.
- Data breach – get a list of hashes, run against common 10,000 password lists for example for initial attempts.
- Bad passwords:
 1. Short
 2. Common
 3. Personal names, confidential personal information
 4. Low diversity in used characters
- Bad password example: password123
- A good password is a password which is random and long.
- Check chosen password with known password dictionaries.

Humans

- Humans are present at some level in every system.
- Every year, an OAIC report is released which breaks down all the reasons for why there are system breaches.
- Technically, every system is breached due to human error.
- The art of deception [Mitnick]; Magicians use misdirection [Let someone focus on something different]
- Spear phishing: target one person to attack/scam/being tricked [looking at the trends that you like/ research about your background/ learn more about you]
- Social engineering is the first tool in the hacker/attacker's plan.
- Hot state: attackers get you into an ad-hoc mental state that is for example urgent to help someone you are familiar with (daughter overseas), losing rationale.

Case study

Harry Houdini was concerned that after his death, mediums would scam his wife by claiming to be able to communicate with him.

Work out a protocol that would solve this problem.

6841 Lecture 3

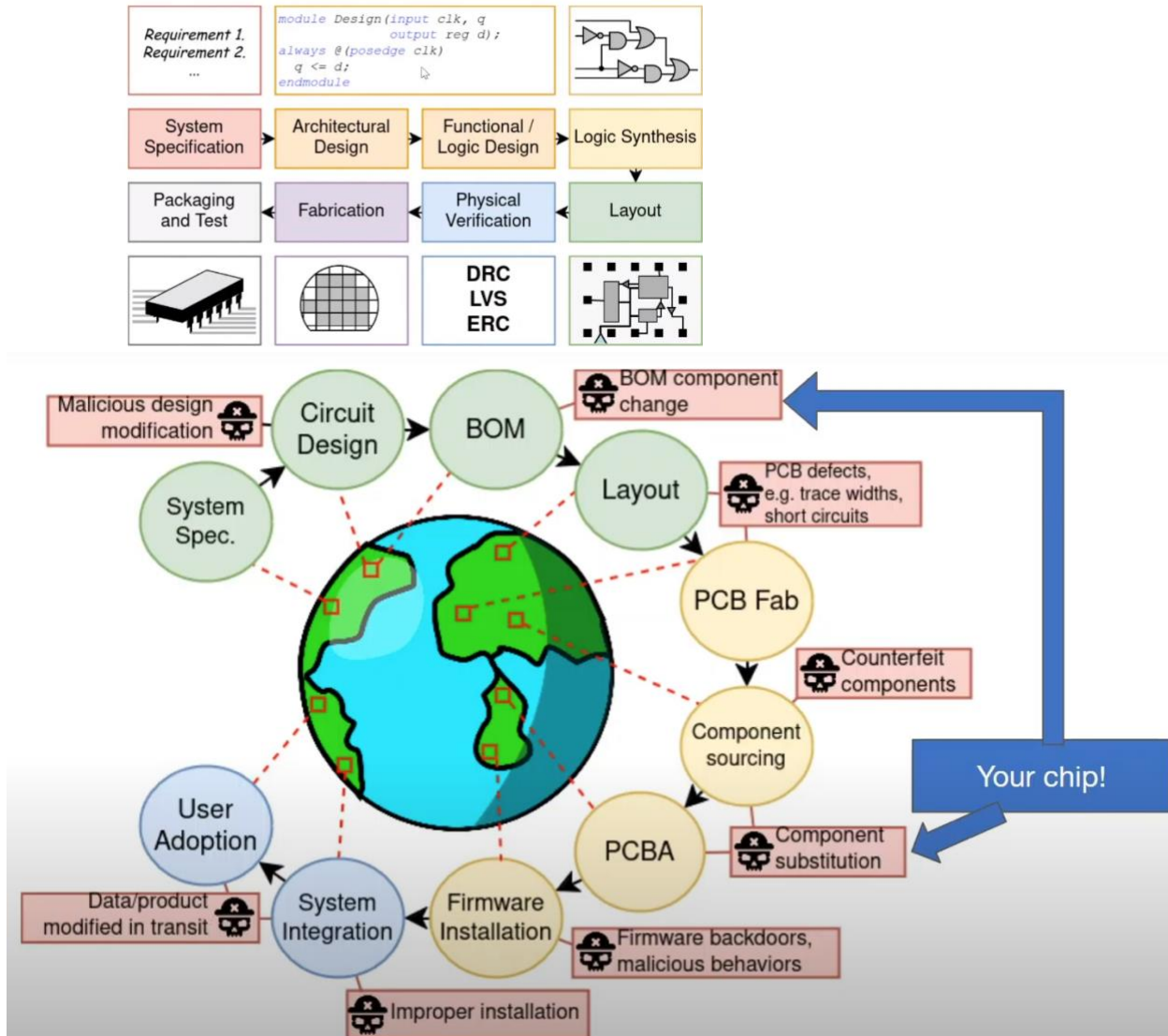
Software security assumes trustworthy hardware.

We trust that:

- 1) hardware executes provided instructions
- 2) hardware does not execute unprovided instructions
- 3) manufacturers build specified circuits
- 4) manufacturers do not build circuits we have not specified

How do we trust that the chip does what we want it to do, and nothing else?

The Integrated Circuit Design Flow



Hardware Risks:

- Hardware Trojans – Similar to software trojan (hidden malicious modifications), but put on the hardware
- <https://spectrum.ieee.org/the-hunt-for-the-kill-switch>
- Supply-chain hardware trojan attack
- Reverse Engineering
 - Used for piracy (to check whether a product is pirated from you, or to copy someone else's)
 - Can help planning/insertion of malicious hardware (but can also help to check whether it's happened)

- IC/IP Piracy and “Overbuilding”
 - e.g. Sinovel wind turbines running stolen AMSC software after AMSC stopped selling to Sinovel
- Counterfeit/forgery
 - Usually taking a part and replacing it with some other part of worse quality, but with all the identifying marks of the legitimate good part
 - Can lead to fatal malfunctions e.g. failure of an ejection seat

Overbuilding - a form of piracy that can occur at an untrusted foundry. A rogue foundry may overproduce extra ICs from the same set of masks and sell them illegally at a cheaper price than the original IC company, thereby increasing the costs for the company only marginally.

3 core requests for hardware security:

1. Protect our designs and the data
2. Identify malicious changes to design
3. Identify exploitable bugs

Ways to prevent hardware attacks

- Cryptography and obfuscation (e.g. cryptographically secure circuits with logic locking, to avoid reverse engineering / overbuilding)
- Logic locking – add some redundant gates – the circuit only works as intended with one set of inputs to the redundant gates
 - How to defeat: there are only so many netlists (possible sets of inputs) - brute force?
- design camouflaging
 - assumes trusted foundry / third party but untrusted user, making design harder to reproduce
 - Downside: takes up more physical space (adding area to gates) and higher power/delay (so to be economically practical, must choose which gates to camouflage)
 - How to defeat/attack: get multiple copies of the circuit, determine list of camouflage gates from the first, test discriminating inputs on the second to quickly identify some gates, then brute force all the other gates
- Side Channels – Record when a CPU is used, and check if other components (Power, Speed, Temp, HPCs, EMI, Vibration, etc.) match a trustworthy CPU
- Functional security analysis and bug-finding, used to detect exploitable mistakes in advance
 - Software equivalent: making design decisions to avoid forming bugs (e.g. never use gets() function, to avoid buffer overflow exploits)

- Assertions and fuzzing (put assertions in, see if anything goes wrong)
- Also formal verification (slow but reliable)

W4 Lecture 1

Supply chain attacks – attack not the business but some other thing in the supply chain

- You're trying to defend yourself, but there are things that feed into you and things that come out of you
 - easier for an attack to attack the things that go into you
 - supply chain attacks e.g.
 - cleaners, infiltrate the cleaning company
 - people that supply you with your hardware
 - outsourcing → expands out risk footprint
 - prevention: do everything yourself (not really feasible)
- Example: sabotaged pagers + Amazon orders for people on the list and get diverted to the US, and NSA puts a keylogger in

Risk:

- Trust exchange: giving the government sensitive data – the issue is the government has a large amount of data. Innovative idea (risky: second mouse gets the cheese)
- Estonian government electronic voting system got hacked
- OceanGate submarine Implosion: low probability high impact risks were ignored due to previous successful missions.
 - Cracking noise during one of the descents was ignored by the owner. Reassurance that something was settling in and everyone accepted it was normal (normalisation of deviance)

Innovation is risky – no established procedures/practices

Risk is invisible: adapt systematic methodology to identify risks

Risks are not always independent.

What are the risks? (important to catch as many as possible)

- One such framework:
 1. Identify the assets (what assets are we trying to protect?, eg. Optus overlooked their main asset – company trust)
 2. Identify sources of attack (e.g. criminals, nation states, human error)
 3. What are all the ways the sources of attack can attack with the power that they have

How can we measure them?

- Likelihood vs impact skew, usually like normal distribution, but... sometimes fat-tail risk
- Fat-tail risk (right-skewed) – low chance of very high impact risk, e.g. Bankstown line still being shut in 2 years, IT projects (usually delayed and cost more). These risks require more care

How do risks interact, chain effects?

- E.g. bushfire -> no radio/mobile signal/wifi -> lack of access to communication
- Should we allow Chinese EV in AU, supply chain arguments?
- You can't treat individual risks separately, you need to think about the correlations between them. Which means you can't multiply small risk high impact events together with the hope of a much smaller risk of a catastrophic event.
- 10 digits of numbers - $10^{10} \approx 2^{30}$ so 30 bits
 - Every 3 0s, 000s = 2^{10}
 - $10^{18} = 18$ 0's for each 3 set of 0 becomes 2^{10} (000,000,000,000, 000, 000) = $(2^{10})^6 = 2^{60}$
 - Reasonableness check

Symmetric ciphers (concluded)

How Symmetric ciphers work

Symmetric cipher uses the same key to encrypt and decrypt a message.

Merkel puzzle is a key distribution algorithm/protocol.

1. A sends a thousand of different keys/different message encrypted with each message with a weak cipher/key to B.
2. B cracks one of the messages to find the key in that message.
3. B tells A what message he/she has discovered to A. A is aware of what key B has now and so
4. A and B use that one for communication.
5. An eavesdropper, C needs to crack all the messages to get the key A and B is using in order to eavesdrop on them. C will eventually get it but it will take time to crack all the messages just to find that particular key that A and B is using currently.

Eg one initial encrypted message could be “hi this is message bghy56!, the key I’m using is sausage”

B cracks that message and says to A “I’m using the key in message bghy56!” eavesdropper can hear this.

A knows it is sausage

Now A and B can have a secure conversation.

All messages sent are moving across an untrusted medium which is why we encrypt them before it is delivered safely to the receiver.

C can also intercept the encrypted message, it would still have to go through all the encrypted messages to find the key since the encrypted messages are essentially not indexed like message 1, message 2, message 3 etc. It can be message aaaaaa, message bjfhfdj, message dkgksdjg, message bghy56! etc.

Key problem: need a huge amount of keys to talk to a lot of people confidentially

- Need roughly $(n^2)/2$ keys for everyone to talk to n people
- Distributing keys is very hard – hard to verify if you've never met in person

Very hard to spot a mistake if you do it yourself – binary search example

Asymmetric Ciphers

- Merkle puzzle – try to solve the problem of key distribution - Imagine A and B wish to communicate securely. A will create a list of puzzles that can be solved by B and sends all of them to B. Each puzzle is encrypted, and the content involves a message number and a password. Once B solves one puzzle, he sends the message number to A and decide to communicate with that password related. So, if anyone else manages to read this message number, it is still safe since they do not know which puzzle it was and which password is related to it.
- - o How can an attacker be kept in the dark if they are the same as a receiver?
 - Need the defender to do way less work than an attacker needs to do
 - o Work ration is squared for merkle puzzle – amount of work the attacker needs to do is the square of the work the defender needs to do
- Different keys used to encrypt and decrypt – if you know the encryption key, it doesn't help you find the decryption key
 - o Eg encrypt with “fish”
 - Receiver tell everyone to send them messages with this key, this is the public key
 - o Decrypt with “chair”
 - Only the receiver knows this key, it is the private key
 - o Anyone can send a message to the receiver with the public key, but only the actual receiver can decrypt it again

- Uses maths!
 - o “one-way” functions like factorising – $p=NP$ problem
 - o We don’t use jumbling/permuting only, since those are pretty easy to crack

Lecture Quiz: Give an example of a correlated risk, where the correlated risk raises the impact.

Week 4 Lecture 2

Dating Scams

- When someone uses a fake profile to gain the trust of a potential romantic partner and then ask for money, gifts, or personal information
- Dating scams works by **exploiting emotional vulnerability**, building trust over time before making a request for money or personal information
 - o People want to believe it’s true, and they unconsciously “conspire” with the scammer to trick themselves, and **you are the best person at tricking yourself**
- Many victims of dating scams **don’t report incidents** due to embarrassment, fear of judgement, or the belief that reporting won’t make a difference

Anti-Phishing Training

- Training programs teach individuals to recognize fake emails, messages, and websites
- Effectiveness of training differs from individual to individual
- UK Police Anti-Fraud Unit – Had a website containing data for the public in order to stop themselves from being scammed
 - o An analysis found that **people that went to the government website for information were more likely to be scammed** than people who didn’t seek any information at all → The website made it worse
 - **Hubris** → People get overconfident

Cognitive Bias

- **Cognitive Bias** – Systematic patterns of deviation from norm or rationality in judgement

- Occur because of the brain's **reliance on heuristics** to make quick decisions, often leading to errors
 - Your brain will trick you over and over again
- **Weight Perception Exercise** – When asked to weigh one matchbox against a stack of three, people often perceive the single box as heavier, even when weights are identical

Refer to Week 3 Lecture 2 for Cognitive Bias Infographic.

Human Weakness and Self-Interest

- **Self-interest** – The act of considering the advantage to yourself when making decisions
- Self-interest often **causes conflicts of interest to arise** → Both parties want to serve their own needs, and neither wants to compromise
- Just human nature but you can't see it (Heroism goes against self-interest and human nature)
- **Prisoner's Dilemma** – A situation where two individuals might not cooperate, even if it would be the best overall outcome, due to the chance that one of them might get to go free
- **Tragedy of the Commons** – A situation where individual self-interest leads to the depletion of shared resources, even though it's in everyone's collective best interest to conserve them
- **Story** – A bartender calling her police fiancé during a rough night at a bar involving Hell's Angels, leading to a whole bunch of cop cars showing up
 - A story about corruption → The bartender had connections with the police, leading to their enthusiastic involvement with the situation
 - Corruption is **not always inherently evil but can still have profound effects on others** → In this case, it potentially diverted police away from people who needed their services to satisfy their own self-interest

Insiders

- **M&M's – An analogy:**
 - Hard on the outside, soft on the inside → Once was the approach to security
 - Can be problematic if insiders are compromised or if there's an inner security flaw
 - People are the weakest link in any security system
 - *You never want a M&M security*
- **Insider** – A person recognized or accepted as a member of a group, category, or organisation
- **Trust is foundational** in society (e.g. road rules) – Breach of trust to us is horrific
 - We get more upset with someone who betrayed our trust, than with someone who has done things that are far worse

- This strong emotional/mental reaction that we have about insiders has two interesting consequences:
 - People often **don't anticipate insider threats**, assuming they are safe → This reluctance to acknowledge such threats can make it **difficult to effectively defend** against them
 - Our reaction to people who do breach trust, AKA **whistleblowers**, is often overly strong
 - E.g. Outrage, going to any lengths to get you (tracking you down, legal actions), attacked physically, professionally and socially (e.g. credibility)

Concepts

Man In The Middle	A cybersecurity threat where an attacker secretly intercepts and possibly alters communication between two parties without their knowledge.
Mig In the Middle	Military variation of a Man-in-the-Middle attack where enemy aircraft relay and reuse identification signals (IFF) from friendly planes to deceive air defences into misidentifying them as allies, bypassing security without breaking encryption.
Proof	<p><i>"To a man with a hammer, everything looks like a nail."</i></p> <p>The concept that even systems mathematically proven to be secure can fail in practice due to overlooked real-world factors, such as incorrect assumptions or errors in implementation.</p> <ul style="list-style-type: none"> • E.g. Mathematical proof might ensure a system follows its specification, but if the specification itself is flawed or doesn't account for human behaviour, the system remains vulnerable.
Tranquility	<p>A flawed security assumption that roles, permissions, or classifications in a system won't change, often leading to vulnerabilities when real-world changes occur.</p> <ul style="list-style-type: none"> • E.g. <i>Bella-LaPadula</i> model
Conflict of Interest	Occurs when personal interests interfere with professional responsibilities, leading to biased or unethical decisions.
Zero Trust	Security model that assumes no inherent trust within the network, requiring verification and monitoring at every step to prevent breaches.

Password Recovery	The most vulnerable part of a password system, as attackers often find it easier to exploit recovery mechanisms (e.g., security questions or email resets) than to compromise the original password itself.
--------------------------	---

W4 Movie: The Truman Show

Spoilers below:

- Dictator director who is a single point of failure. One possible alternative is a committee consensus should override him?
- Who / what do you trust? What if everyone are actors? Best friend saying "If everyone else is in, I'm in" - you believe the world as its presented
- Its hard to be the sole outlier and disagree with everyone else
- Human psychology is hard to predict, but also somewhat predictable
- To reduce the risk of him escaping the house, they should have more people (or nowadays AI) looking at the cameras near the house. Very unlikely risk but has severe consequences, so people became complacent. The lead organisers incorrectly assumed tranquility, and that if he's sleeping, he wouldn't be escaping, they were complacent.
- Actors looked into the camera, breaking the rule of cinematography
- Privacy
- Conflict of interest - duty to the kid, but also the money raised
- Insider attack: Marlon breaking his trust, lying to his face – society normalising suspicious things, such as the personal radio the - eg hitler youth, frog boiling in a pot
- The director orchestrated phobia of water
- The directors had confirmation bias, wanting him to go back to normal. He went back to typical behaviours to lull them into a false sense of security, cognitive biases

- Human Error: an actor from far away from the small town said "your welcome, truman", how did they know his name?
- Never waste a good crisis – his wife ‘dying’ so he moved into the basement
- Too many extras, hard to coordinate none of them leaking information to Truman – the counterpoint was the very small number of leaks by the NSA, something similar could be used by the movie crew.

6841 Lecture 4 – JavaScript Injection (XSS)

Admin

- M1 Macs are ARM, they won't work with the buffer overflows. GitHub Student will give you \$200 credit for Digital Ocean Droplet.
- The SecLab is K17 room G11, big glass room past the male bathroom. Should be covered in SecSoc Posters.
- PUT doesn't properly exist in SQL, only really in HTTP. Heavily defined by individual devs. Just use INSERT.
- Exam is likely going to be a take home exam on your own hardware. (Important if you are using a new chipset Mac)

Ethics

- Don't drop tables.
- Don't set fake flags.
- Don't use tools you don't understand like SQLMap.
- **Don't be a dick.**

What can you do with XSS

- Steal sessions (cookies)
- Credential theft
- Create a Botnet
- Mine Crypto (it's going up even more this time)
- AdSense fraud
- Almost anything that you can do in JS can be done via XSS

Tangent: Cookies are useful but a complete nightmare for privacy. Mainly third-party cookies as they can collect immense amounts of data.

Examples include Facebook knowing if you are part of the LGBTQI+ community 2 years before you come out (on average) and Target knowing a 17 year old was pregnant before she knew.

What is JavaScript?

JavaScript is the language of dynamic function for webapps. It sucks but hey you can move things and change colours. It has “loose” type conventions which can cause some nonsensical results

What is XSS?

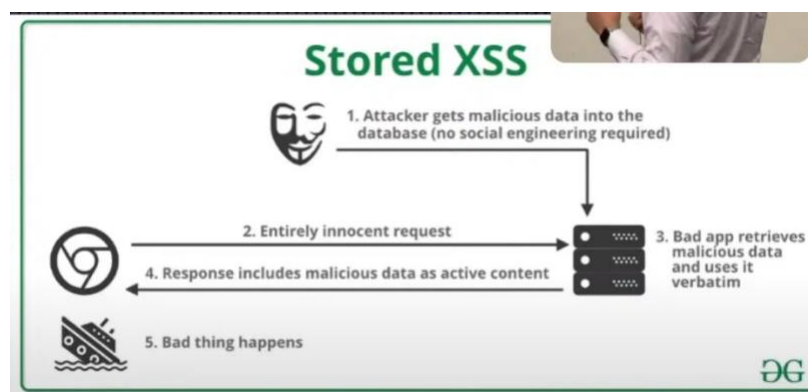
Originally called Cross Site Scripting as you could run scripts from one site to run on another. It now includes direct injection of scripts (Hence **JavaScript Injection**). XSS is a vulnerability that allows an attacker to run unauthorized code on a client's web browser as if it was from the website.

Notable examples of XSS Exploits

- Google Search
- Counter Strike panorama XSS (Via the vote kick prompt in frontend) (twice)
- Self-retweeting tweet (Tom Scott “How The Self Retweeting Tweet Worked” video)
 - TweetDeck didn't sanitize the `<script>` HTML tag.
 - Exploit forced the site to click the retweet button and the retweet confirmation button all in 140 characters.
 - An example of a second order attack (Hit Twitter users through TweetDeck).

Types of XSS:

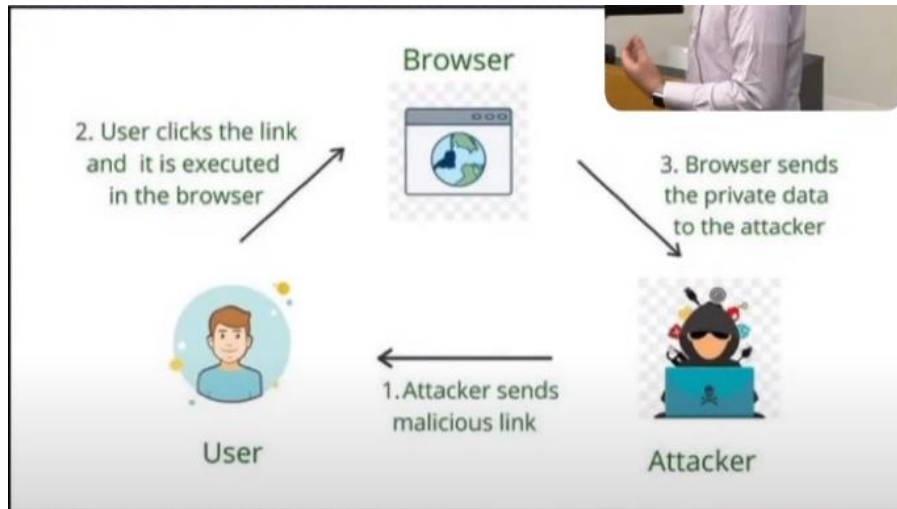
Stored (P1)



1. Attacker stores malicious data into a webserver database
2. User requests data (like normal)

3. Database sends malicious data
4. Malicious data runs on user's site

Reflected (P1 / P2)



1. Attacker sends malicious link to user
2. User clicks bad link, and it gets run in the browser
3. Browser sends stolen data back to attacker

A very common attack, mostly done via phishing. Often done to steal sessions via cookies.

Self

- Type of reflected XSS that can only affect yourself.
- Not super useful but can mean proper framework/ procedure is not being followed. There may be a vulnerability somewhere else on the application.

DOM (Document Object Model) (NOT ASSESSED)

- Similar to Reflected but attacker writes to the DOM rather than returning payload

DOM BASED XSS VERSUS REFLECTED XSS	
DOM BASED XSS	REFLECTED XSS
An advanced type of XSS that occurs by writing data to the Document Object Model (DOM)	Second and the most common type XSS in which the attacker's payload is a part of the request that is sent to the web server
Occurs by processing data from an untrusted source by writing data to a potentially dangerous sink within the DOM	Occurs when an application obtains data in an HTTP request and includes that data within the immediate response in an unsafe way
More complex than reflected XSS attacks	Simpler than DOM-based XSS <small>Visit www.PEDIAA.com</small>

Example code

```
Example Vulnerable Code

<?php
    $searchQuery = $_POST['user_id'];
    $query = "SELECT * FROM articles WHERE title LIKE '%searchQuery%'";
    $results = database.execute($query);
?>

...

<h3>You searched for <?php echo $searchquery; ?></h3>

<?php
    for ($i = 0; $i < len($results); $i++) {
        echo "<b>$results[$i][0]</b>";
        echo "<p>$results[$i][1]</p><br>";
    }
?>
```

Note: php sucks but lots of thing use it because it works with HTML

\$searchQuery is taken directly from the user input with no sanitization

Demo at <https://xss-game.appspot.com/> (look this up don't click links)

<script> is not the only way to run JavaScript, you can also use some HTML tags

[https://docs.google.com/document/d/1YwuY6j1QnjITk_Ew-](https://docs.google.com/document/d/1YwuY6j1QnjITk_Ew-HmsdCNkWBWhNDqrILKxphSEj6I/edit?usp=mail_thread&ts=66fcad3d)

[HmsdCNkWBWhNDqrILKxphSEj6I/edit?usp=mail_thread&ts=66fcad3d](https://docs.google.com/document/d/1YwuY6j1QnjITk_Ew-HmsdCNkWBWhNDqrILKxphSEj6I/edit?usp=mail_thread&ts=66fcad3d)

Image x does not exist so it will run on error which has action alert(1)

Can do this for any event: like ondragstart or even onload (with an image that does exist)

Challenges this week: Stealing cookies

- Use JavaScript to send the cookie somewhere else
- You can use the browser console to run JavaScript and craft a payload
- If you make the blog load into a different site, then just logout and create a new login
- Use RequestBin to find the cookie once you redirect the admin

Week 5 Lecture 1 (Podcast)

The Fundamental Problem of Security

- *"The problem of dreaming"*
- **Computers live in a different world than we live in** → They operate on data and logical information, whereas humans perceive the world through physical and sensory experiences
 - The **disconnect between these "worlds"** creates fundamental challenges in ensuring security
- We might think that they are able to interact with the real world due to its interfaces or transducers but we need to understand that **transducers or interfaces are not an inherent part of the computer itself**
 - **Transducer** – Translates logical data (like a binary instruction) into physical signals or actions that can interact with the real world
 - These devices or interfaces are **"bridges"** because they transform what the computer understands (logical, abstract data) into something tangible or perceivable in the real world
 - The computer itself **doesn't inherently "understand" or "exist" in the physical realm**; it relies on these interfaces to do so
- **Analogy** – The computer is a prisoner in a room entirely closed off from the outside world
 - They receive notes from the outside world, their mother, via a guard
 - The prisoner cannot confirm if the messages truly come from their mother, if she's really there since there are no side channels, or if the guard fabricated them

- *How can we, as the computer, trust the data we receive or ensure it hasn't been intercepted or impersonated?*

Why Is This A Problem?

- Computers are **great at the "CI" of "CIA"**, but **authentication is about the outside world**
- We are asking the computer to make decisions about the real world, which fundamentally, the **computer cannot do (at least not perfectly)**
 - All it can do is compare with the information it has been given and the one in their knowledge (e.g. fingerprint)
- **A computer has no knowledge of the outside world** – A computer simulation can run an old Atari game, and the program wouldn't "know" it's running on a different system
 - This means they are **vulnerable to deception** if the input data is manipulated (e.g., phishing, counterfeit signals, or man-in-the-middle attacks)

How do we know we're not living in a simulation?

- **Argument by stone** – Dismissing a claim as absurd without providing evidence or addressing its reasoning
- Computers can only "see" and process the inputs given to them
 - If the inputs are fake, the computer can't inherently detect the deception (e.g., fake credentials or a spoofed signal).
- Similarly, humans rely on sensory and logical inputs to perceive the world
 - If these inputs are manipulated, we cannot directly detect it
 - If we can't detect them, how do we prove that they're not real?

- The computers will do their best with the data they receive, but if the data is fake then → garbage in = garbage out
- **The Paradox of Communication** – Two generals on opposite sides of a valley need to coordinate an attack but can only communicate via unreliable messengers (e.g. pigeons, which have a 95% probability of reaching the other general)
 - Shows difficulty of ensuring secure communication when the channel is untrustworthy

WE CAN'T SOLVE THE AUTHENTICATION PROBLEM

Authentication

- **Authentication is knowing the “who”** → Focuses on verifying the entity behind a request

- *How do you know if the person in front of you is your partner?*

- Really hard to analyse things that are obvious (thinking “Yeah, I know it all”)
- A lot of it is what you’re expecting and a lot of it is based on visual data
 - **Something you know** – Middle name, information you share
 - **Something you have** – Clothes, jewellery
 - **Something you are / Something you can do** – Gait, height, ability to whistle

- **Computer Equivalent:**

- **Something you know** – Passwords, PINs, security questions
- **Something you have** – Physical tokens, access cards
- **Something you are** – Biometrics, face recognition
 - Computers mostly authenticate based on physical factors

- Ultimately they are all the same thing – **Something you know**
 - Every form of authentication that a computer does is a **shared secret**

Problems in Authentication

- Each method of authentication **has vulnerabilities**:
 - Passwords can be guessed or stolen
 - Tokens can be lost or forged
 - Biometrics are difficult to change and can be spoofed (e.g., fake fingerprints)
- Combining two or more authentication factors (e.g. password + fingerprint) improves security, but **MFA isn’t foolproof** → Creates a **false sense of security** (similar to security by obscurity), may create **usability issues**, and if implemented poorly, it **can introduce new vulnerabilities**
- Can also become **annoying to authenticate every time** → Becomes a user and usability problem if you have to login every time you do something
 - **Compromise** – If someone has been working for ten minutes on an account, they don't have to reauthenticate until after they logged out or have been idle for 20 minutes, etc.
 - New problem → **Opens an attack window**
- The **more elaborate and onerous** the authentication, the **greater the type II error chance**
 - **Need for tolerance** – Need to be able to forgive lots of small differences if we have to authenticate using something you are, since people change from day to day

- **Tolerance is a problem** → Silent invisible failure

Authentication Protocols

- We use authentication to **solve practical real world problems**
 - E.g. *"I would like to make sure I am the only person allowed to access my bank account"* (high level objective, not specific)
 - There are two ways it can fail → Type I and type II
 - Improving one error rate often worsens the other
- How often should authentication occur? After every keystroke? Only once?
 - In practice, practical problems that arise with it happening too often is **very visible** to the user, as it takes up their energy and time
 - That **visibility is a curse**:
 - Encourages us to come up with solutions where authentication happens once and then afterwards weaker things happen, assuming that that person is still there (authorisation)
 - If something is visible, then people can sell sometimes ineffective security products for it (**security theatre**)
 - Increases complexity, creating false sense of security
 - However, authentication problems are **not guaranteed to be visible and fail visibly**
- Computers don't truly "know" who you are – Authentication typically **grants access to a digital identity**, which may or may not correspond to the correct person
 - There's often **no direct link** between a person and their digital identity → Gap creates vulnerabilities (**not end-to-end**)
- *So how do we authenticate securely in digital spaces, especially over untrusted networks?*

S/Key Algorithm

- **Concept** – Two parties **agree on a secret beforehand**, assuming that other people don't know it (e.g. "What did I have for dinner last night?")
 - Problem with the key is that we **can't ever use it again** because we don't know if someone has copied it or overheard it when exchanging it
- What we really need is a long sequence of keys you can use, and once it has been used, should be able to throw it away and use a different key while the server keeps track of which ones you have or haven't used
- **S/Key Algorithm** – Uses a one-time password system derived from a **sequence of hashed keys**

- **Process:**
 1. A shared key is repeatedly hashed (e.g. 10 times)
 2. The last hash (10th) is stored on the server
 3. Each time you log in, you use the next-to-last hash. The server verifies it by hashing it once
 4. Once used, the hash is discarded, and you use the one before that the next time you log in
- **Advantage** – Even if someone intercepts a hash, it cannot be reversed to generate previous keys
- **Limitations** – Eventually, you run out of hashes and must reinitialise

Challenge Response

- **Concept** – Instead of sharing a secret, the **system sends a different challenge each time** (e.g., "Prove you know the secret without revealing it")
- The user responds with a calculated answer based on the secret
- **Advantage:**
 - Prevents **replay attacks** (where an attacker reuses old responses)
 - Great way to detect "liveness" → That entity is really there

HOTP (HMAC-based One-Time Password)

- **HOTP** generates one-time passwords **using a counter as an input**
- **Involves:**
 - **A shared secret** – A unique key that both the server and the user know (e.g. stored in an app like Google Authenticator)
 - **A counter** – A number that increments with every authentication attempt
- **Process:**
 - The counter and the shared secret are combined and passed into an HMAC (Hash-based Message Authentication Code) function
 - The HMAC function generates a hash, which is then converted into a short, human-readable numeric password (e.g. 6 digits)
 - The server and the client keep track of the same counter
 - When the user enters their OTP, the server computes the expected OTP using the counter and the shared secret
 - If it matches, the user is authenticated
- **Strength:**
 - HOTP can generate OTPs indefinitely, as the counter can increment forever.
- **Weakness:**
 - Replay attacks are possible if the same OTP is reused

- If the counter becomes out of sync (e.g., if a user generates multiple OTPs without using them), the system must implement a "look-ahead" mechanism to check for valid OTPs
- **Example:**
 - Counter – 1001
 - Shared secret – ABC123
 - HMAC(ABC123, 1001) → Hash → OTP: 894763

TOTP (Time-based One-Time Password)

- **TOTP builds upon HOTP** but **uses time instead of a counter**
 - It ensures that passwords are valid only for a specific time interval
- **Inputs:**
 - A shared secret – Same as HOTP
 - The current time – Divided into fixed intervals (e.g, 30 seconds)
- **Process:**
 - The current time is divided by the interval size (e.g. Unix time / 30 seconds)
 - The result is fed into the HMAC function along with the shared secret
 - The output is converted into a short numeric OTP
 - Both the server and the client compute the OTP using the current time and the shared secret
 - If they match, the user is authenticated
- **Strength:**
 - OTPs expire quickly, so even if an attacker intercepts one, it becomes useless after the time interval
 - No need to track a counter, reducing the chance of synchronisation issues
- **Weakness:**
 - Vulnerable to interception during the validity period (e.g., a 30-second window)
- **Example:**
 - Current time – 1674259200 seconds (Unix time)
 - Interval size – 30 seconds
 - Time interval – $1674259200 / 30 = 55808640$
 - HMAC(ABC123, 55808640) → Hash → OTP: 273482

Differences Between HOTP and TOTP

| Feature | HOTP | TOTP |
|-----------------|------------------------------------|------------------------------------|
| Input | Counter | Time (e.g., 30-second intervals) |
| Validity | Does not expire until used | Expires after the time interval |
| Synchronization | Requires counter synchronization | Requires time synchronization |
| Use Case | Good for infrequent authentication | Better for frequent authentication |

RSA (Naive)

- **RSA** – An **asymmetric encryption algorithm** that uses a pair of keys:
 - Public key – Used to encrypt data (known to everyone)
 - Private key – Used to decrypt data (kept secret)
- **Process:**
 1. **Key Generation:**
 - Two large prime numbers are chosen and multiplied to create a modulus
 - A public key and a private key are mathematically generated from these numbers
 2. **Encryption:**
 - A sender encrypts a message using the recipient's public key
 - The encrypted message can only be decrypted by the recipient's private key
 3. **Decryption:**
 - The recipient uses their private key to decrypt the message
- RSA can **prove identity** by signing messages:
 - A user encrypts a message (or hash) with their private key (digital signature)
 - The recipient verifies it by decrypting it with the user's public key
- **Weakness:**
 - It is computationally intensive
 - Messages must be small because of the modulus size
 - Modern systems often combine RSA with hashing (e.g. signing a hash instead of raw data) for efficiency
- **In practice:**
 - Most people use RSA to encrypt, then **combine it with a nonce**
 - **Nonce** – A **random number or value** that is used only once in a communication session
 - Helps **prevent replay attacks** (where an attacker reuses a valid message to fool the system)

- This ensures that even if the same message is encrypted multiple times, the result will always be different due to the nonce
- The **message + nonce might be salted and hashed**
 - **Salted hash** – When a salt (random value) is added to the data before it is hashed
 - Makes it harder for attackers to precompute hash values (rainbow table attacks)
 - Ensures that identical data results in different hash outputs because the salt is unique
 - **RSA has a size limit** – To address this, the message is hashed (hashes are fixed size regardless of input size)
 - Hashing condenses the message into a smaller, secure representation

RSA Signing Process (Simplified)

- **Sender's Side:**
 - Original Message: "Hello World!"
 - Compute a hash of the message (e.g., SHA-256)
 - Hash – f572d396fae9206628714fb2ce00f72e94f2258f
 - Encrypt the hash with their private RSA key (this is the digital signature)
- **Receiver's Side:**
 - Decrypt the signature using the sender's public RSA key to get the hash
 - Hash the received message and compare it to the decrypted hash
 - If they match, the message is authentic and unaltered

OAuth (Open Authorisation)

Once authenticated, how does the system ensure ongoing access is valid?

- **Solution** – Use **authorisation tokens**
 - After initial authentication, a token is generated and passed with requests
 - Similar to Single Sign-On (SSO), where users authenticate once and use the token for continued access
- **Strengths** – Convenient for users
- **Weaknesses** – Tokens can be hijacked if intercepted (e.g. Man-in-the-Browser attacks)
- OAuth typically involves the following steps:
 1. **Authentication:**
 - You log in to the resource provider (e.g. Google, Facebook)
 2. **Authorization Grant:**

- The resource provider asks you for permission to let a third-party app (e.g. a fitness tracker) access certain resources (e.g. your email address)

3. Access Token:

- This token allows the app to perform actions or fetch data without needing your credentials

4. Access Resources:

- **Key Terms:**

- **Access Token** – A string used to access resources.
- **Refresh Token** – A longer-lived token to get a new access token when the old one expires.

Attacking MFA

- Common Attack Vectors:
 - **Social Engineering** – Convince support to reset a password or provide access, or phishing
 - **Replay Attacks** – Reuse old authentication data if not properly secured.
 - **Session Hijacking** – Steal tokens through malware or browser-level attacks
 - **Man-in-the-Middle** – Intercepting data during transmission
 - **Compromising Devices** – Physically stealing or photographing MFA devices like RSA SecureID tokens
- MFA can be vulnerable since it often relies on external providers or devices that may not be robustly designed
 - A failure in one part of the system can compromise all accounts relying on it

W5 – Tuesday

Humans and insiders - The double-edged sword of whistleblowers and humans in the loop

- very prone to single point of failure risks
- people often react very strongly
- having humans makes you more prone to type I and type II errors

argument by stone when you want to show something is ridiculous but you don't

CIA TRIAD – confidentiality, integrity, authentication

- first two are about the data and the last is about the sender (the real world)

- the fundamental problem with computer security: the computer knows nothing about the real world and is only working based on inputs and data it is fed
 - o if you put a brain in jar and give it enough data it won't know it's not really living

Issues with privacy:

- Very tricky
- Data wants be free – it's a self-interest maximizing argument (people wouldn't want their data to be free but other's data can help them)
- Hard because – there's no going back, can't explain to people you want to be private
- Strongly related to your quality of life –higher stress levels when observed by others
- Can you trust the government?
- Idc about privacy because I have nothing to hide – laws could change and you could become a criminal suddenly (it was illegal to serve black people before)
- Tranquility – future change

Authentication (short context for law people)

- Authentication – something you know, something you have, something you are
- Authentication has to do with humans whereas confidentiality and integrity refer to the data.
- The ultimate problem in security as that computer chips (decision maker) have no idea how the real world works.
 - o Password? Someone could have found out the password by themselves
 - o Biometric? Fake thumbprint
- The point is, the computer just acts based on the data it gets fed. If a person is highly motivated to trick the system, we do not really have any way of knowing.
- The person/computer making the decision knows nothing about the outside world. It just gets data that summarises the outside world.
- **Request:** Did someone hear what the poster winner said towards the start of the lecture? If so please write it here and delete this sentence please (I couldn't hear it online with the echo).
- Challenge of authentication, we don't know how to do it (connect physical and digital world for authentication)
- Anything that attempts to identify or authenticate you can be ultimately flawed.
- Everything in the real world can be forged or faked. How can we make certain that someone's identity can be trusted from data?

- Might be impossible to perfectly authenticate but we can make it hard/expensive to fake factors that we use to authenticate.

Privacy

Privacy and security are highly related issues.

Three people affected by privacy.

- Government – want your data for honourable and good services
- Companies – want to sell you stuff/make money
- You – want your privacy

People don't value privacy enough (not a self-balancing system).

Once the data is collected there's no going back

Privacy can lead to being compromised e.g. personal information is often used to answer security questions.

In Australia, we tend to use the term “privacy” in relation to data protection. The International Human Rights has some clauses about privacy, but there is nothing in the Australian constitution.

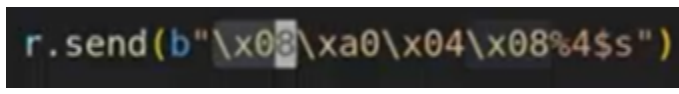
Right now, you cannot sue anyone for an invasion of privacy, but there is a new bill that may or may not be passed.

- Human Factors in Security: Emphasize the dual role of humans in security systems, acting as both safeguards and potential sources of error.
- Context and Background: Reference the protest during the Ukraine war, illustrating media manipulation and its relevance to trust and authenticity in information.
- Insider Threats and Whistleblowing: Highlight the exploration of trust issues and real-world implications of insider threats, drawing parallels with the challenges faced by whistleblowers in legal contexts.
- Challenges in Authentication: Expand on the difficulties of traditional authentication methods (e.g., passwords, biometrics) being flawed and how they cannot reliably discern real identities.
- Digital Identity and Its Limitations: Add the perspective that digital identity systems struggle with verifying authenticity, a key issue in the discussion on privacy.

- Concluding Thoughts on Authentication and Privacy: Stress that there is no foolproof method for authenticating identities in the digital realm, linking back to privacy as an integral aspect of security.
- Public Sentiment on Privacy: Acknowledge the growing societal recognition of privacy's importance and that attacks on privacy are increasingly seen as serious crimes.
- Cultural Perspectives on Privacy: how historical abuses in Europe have led to a more cautious approach to data privacy, contrasting with Australian attitudes that may lead to complacency.

6841 Lecture 5 – Format String Vulnerabilities

Essentially, we are exploiting printf to leak vulnerabilities



`r.send(b'\x03\xa0\x04\x08%4$s')` (little endian)

The printf() family produce output according to a specific format.

With printf, we have specifiers (think %d, %s, %x, %n).

%n – write the number of characters written so far into a variable (!!!).

History

- A class of vulnerability that was disclosed in the early 2000's.
- Was previously known but considered to be harmless.
- They are kind of a big deal (remote code execution).
- Easy to find: `grep ".*printf([^\"]\')"`

Grep `".*printf([^\"]\')"`

Printf is a variadic function – e.g. main (will take either 2 or no arguments).

Printf will take from 1 to an indefinite number of arguments and the format-string specifies how many arguments there are.

If we have too few arguments, that will lead to an out-of-bounds read.

Typically, the stack is kept quite far away (address-wise) from data memory

| Dec | Hx | Oct | Char | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|-----|----|-----|------------------------------------|-----|----|-----|-------|--------------|-----|----|-----|-------|----------|-----|----|-----|--------|------------|
| 0 | 0 | 000 | NUL (null) | 32 | 20 | 040 | | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 1 | 1 | 001 | SOH (start of heading) | 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 2 | 2 | 002 | STX (start of text) | 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 3 | 3 | 003 | ETX (end of text) | 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 4 | 4 | 004 | EOT (end of transmission) | 36 | 24 | 044 | $ | \$ | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 5 | 5 | 005 | ENQ (enquiry) | 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |
| 6 | 6 | 006 | ACK (acknowledge) | 38 | 26 | 046 | & | & | 70 | 46 | 106 | F | F | 102 | 66 | 146 | f | f |
| 7 | 7 | 007 | BEL (bell) | 39 | 27 | 047 | ' | ' | 71 | 47 | 107 | G | G | 103 | 67 | 147 | g | g |
| 8 | 8 | 010 | BS (backspace) | 40 | 28 | 050 | (| (| 72 | 48 | 110 | H | H | 104 | 68 | 150 | h | h |
| 9 | 9 | 011 | TAB (horizontal tab) | 41 | 29 | 051 |) |) | 73 | 49 | 111 | I | I | 105 | 69 | 151 | i | i |
| 10 | A | 012 | LF (NL line feed, new line) | 42 | 2A | 052 | * | * | 74 | 4A | 112 | J | J | 106 | 6A | 152 | j | j |
| 11 | B | 013 | VT (vertical tab) | 43 | 2B | 053 | + | + | 75 | 4B | 113 | K | K | 107 | 6B | 153 | k | k |
| 12 | C | 014 | FF (NP form feed, new page) | 44 | 2C | 054 | , | , | 76 | 4C | 114 | L | L | 108 | 6C | 154 | l | l |
| 13 | D | 015 | CR (carriage return) | 45 | 2D | 055 | - | - | 77 | 4D | 115 | M | M | 109 | 6D | 155 | m | m |
| 14 | E | 016 | SO (shift out) | 46 | 2E | 056 | . | . | 78 | 4E | 116 | N | N | 110 | 6E | 156 | n | n |
| 15 | F | 017 | SI (shift in) | 47 | 2F | 057 | / | / | 79 | 4F | 117 | O | O | 111 | 6F | 157 | o | o |
| 16 | 10 | 020 | DLE (data link escape) | 48 | 30 | 060 | 0 | 0 | 80 | 50 | 120 | P | P | 112 | 70 | 160 | p | p |
| 17 | 11 | 021 | DC1 (device control 1) | 49 | 31 | 061 | 1 | 1 | 81 | 51 | 121 | Q | Q | 113 | 71 | 161 | q | q |
| 18 | 12 | 022 | DC2 (device control 2) | 50 | 32 | 062 | 2 | 2 | 82 | 52 | 122 | R | R | 114 | 72 | 162 | r | r |
| 19 | 13 | 023 | DC3 (device control 3) | 51 | 33 | 063 | 3 | 3 | 83 | 53 | 123 | S | S | 115 | 73 | 163 | s | s |
| 20 | 14 | 024 | DC4 (device control 4) | 52 | 34 | 064 | 4 | 4 | 84 | 54 | 124 | T | T | 116 | 74 | 164 | t | t |
| 21 | 15 | 025 | NAK (negative acknowledge) | 53 | 35 | 065 | 5 | 5 | 85 | 55 | 125 | U | U | 117 | 75 | 165 | u | u |
| 22 | 16 | 026 | SYN (synchronous idle) | 54 | 36 | 066 | 6 | 6 | 86 | 56 | 126 | V | V | 118 | 76 | 166 | v | v |
| 23 | 17 | 027 | ETB (end of trans. block) | 55 | 37 | 067 | 7 | 7 | 87 | 57 | 127 | W | W | 119 | 77 | 167 | w | w |
| 24 | 18 | 030 | CAN (cancel) | 56 | 38 | 070 | 8 | 8 | 88 | 58 | 130 | X | X | 120 | 78 | 170 | x | x |
| 25 | 19 | 031 | EM (end of medium) | 57 | 39 | 071 | 9 | 9 | 89 | 59 | 131 | Y | Y | 121 | 79 | 171 | y | y |
| 26 | 1A | 032 | SUB (substitute) | 58 | 3A | 072 | : | : | 90 | 5A | 132 | Z | Z | 122 | 7A | 172 | z | z |
| 27 | 1B | 033 | ESC (escape) | 59 | 3B | 073 | ; | ; | 91 | 5B | 133 | [| [| 123 | 7B | 173 | { | { |
| 28 | 1C | 034 | FS (file separator) | 60 | 3C | 074 | < | < | 92 | 5C | 134 | \ | \ | 124 | 7C | 174 | | | |
| 29 | 1D | 035 | GS (group separator) | 61 | 3D | 075 | = | = | 93 | 5D | 135 |] |] | 125 | 7D | 175 | } | } |
| 30 | 1E | 036 | RS (record separator) | 62 | 3E | 076 | > | > | 94 | 5E | 136 | ^ | ^ | 126 | 7E | 176 | ~ | ~ |
| 31 | 1F | 037 | US (unit separator) | 63 | 3F | 077 | ? | ? | 95 | 5F | 137 | _ | _ | 127 | 7F | 177 | | DEL |

Source: www.LookupTables.com

Demo

We aim to find the base pointer because that is where the data we need is stored

'%#x' adds the '0x' prefix

```
Hi, what's your name?
200 f7e365c0 80491e0 100 0 25207825 78252078 20782520 25207825 78252078 20
782520 25207825 78252078 20782520 25207825 78252078
Give me your favorite number :)
```

'\$' Index into the arguments - one indexed – starting from the start of the buffer and going 4 bytes at a time

b'%2\$s' - prints the second argument

```
solve.py ex1  solve.py ex2 X  example2.c
ex2 > solve.py > ...
2
3  r = process("./example2")
4
5  r.recvline()
6
7  payload = b"%2$s"
8
9  #printf("%x %x %x %2$x", arg1, arg2, arg3, arg4)
10
11  r.sendline(payload)
12
```

Reading the above output, we see that the 6th argument is where printf starts printing its own input – “%#x” a whole bunch of times

Remember that arguments are the last things pushed to the stack before a function call

```
read(fd, &secret_number, 4);

0x08049213 <+77>:  sub    esp,0x4
0x08049216 <+80>:  push   0x4
0x08049218 <+82>:  lea    eax,[ebx+0x38]
0x0804921e <+88>:  push   eax
0x0804921f <+89>:  push   DWORD PTR [ebp-0xc]
0x08049222 <+92>:  call   0x8049040 <read@plt>
```

x/x \$eax will give us the address of eax

```
pwndbg> p $eax
$1 = 134529068
pwndbg> x/wx $eax
0x804c02c <secret_number>: 0x00000000
```

If your secret is not in the main function, you need to track down which function assigns a value to it

Having the address, we can then print the value in different formats using the format specifiers e.g. “%s”

Input a pointer -> dereference that pointer

```
secret_number_addr = 0x804c02c
another_secret_number_addr = 0xffffcdd8

>>> hex(0xcdd8-0xcdd8)
'0x200'
>>> 0x200/4
128.0
```

6 lots of 4 bytes to the start of the buffer + 128 lots of 4 bytes from the start of the buffer to address of secret_number

What can be leaked?

Leaking pointers


Leak variables

Dump memory

Gotchas

Misaligned stack addresses

Reading multiple values at once

Here is a giant pdf on format stringys I found online if anyone is interested in reading it 
chapter 3 in particular is quite useful:

<https://cs155.stanford.edu/papers/formatstring-1.2.pdf>

W7 – Monday

Integrity

CIA Principles

- Confidentiality: maintain the secrecy of information
- Integrity: transmit info without it changing or degrading
- Authentication... to be discussed in future lectures

Integrity: Being able to verify that the information has not changed.

We cannot trust any message if integrity isn't maintained, even if we have authenticated it.

Data should be:

- complete - full form, no elements missing, truncated, or filtered
- accurate - isn't aggregated in any way (e.g. rounding up/down)
- consistent - remains unchanged regardless of how/how often accessed and how/how long stored

The Luhn Algorithm

Credit card numbers are generated algorithmically.

Every bank has a unique set of starting numbers.

The bank then generates the rest of the 15 numbers, and the final digit is generated via the Luhn Algorithm.

Checksum - a way of validating something by storing data into the data itself. This makes it useful for checking data integrity

Credit Card Security

- First 4-6 digits unique to your bank
- last 4 digits are not obscured at all and are usually plainly printed, such as on your receipt "*****4826"

- And the last digit is a checksum
- Banks also do not need the 3 digit security code on the back, it is mostly for merchants since it costs them money to complete a transaction without it.

Checksums

- Used to validate what has been sent
- powerful but not all knowing
 - E.g. I send a TCP packet and at the end I send a 1 or a 0 that indicated how many 1 bits are in my message - even number I send 0 and odd number I send 1.
 - Can detect if a bit has been changed but not if two bits have changed

Blockchain

- Not inherently integral since attacks can still occur on blockchains
- 51% attack has the highest impact: Someone with 51% of the processing power can completely rewrite the blockchain piece by piece

Types of Hashes

| Cryptographic | Non-cryptographic |
|--|---|
| <ul style="list-style-type: none"> • Need to be resistant to certain types of attacks • Slow by design • Examples: MD5, SHA1/2/3 • Try to minimise which values get mapped to the same | <ul style="list-style-type: none"> • Need to be fast and not necessarily unique every time • Example: Creating a hash table in python. If a value is already taken in the table, just use the next cell.. |

We will solely be talking about cryptographic hashes from here on out.

Hash – variable length put into a fixed size meaning some things will have to be mapped to the same hash since we have infinite set of inputs and a fixed set of outputs.

Use cases:

- Passwords
- Digital signatures
- Integrity checks

```
# terminal automatically adds a new line
echo "password" | md5
# -n removes the new line
echo -n "password" | md5
```

If you were sending a message, you could send the hash of the message along with it as an integrity check. The receiver can perform the hash again and check it has not been tampered with.

However, since data and control are in the same message, a hacker could intercept it and change both the message and the hash. This is why we need some sort of secret key between the sender and receiver that cannot be tampered with. This is known as a **Message Authentication Code (MAC)**, and the most common implementation is Hash MAC (HMAC).

Attacks on hashes

- Brute force – go through a whole bunch of common passwords and check for a hash collision
 - More difficult if hashing takes a long time – even 100ms to compute a hash = 11 days to brute force 10 million passwords
- Collision attack
 - Eg: In the bank example, you would find 2 numbers that hash into the same thing, go to one bank and withdraw the higher amount, send a message with the lower amount, then go to a different bank and balance your accounts.
- Pre-image attack
- Second pre-image attack – given a plaintext and a hash, look for a different plaintext that would result in the same hash

Attacks on Hashing

Overview

- Collision attack - Find two plaintext that hash to the same thing.
 - Done in square root of output set size.
- Pre-image attack - Given a hash, find ANY plain text that hashes to the given hash.
 - Done in half the output set size.
- Second Pre-image attack - Given a plaintext and the hash of the given plaintext, find ANOTHER plaintext that hashes to the given hash.
 - Also done in half the output set size.

Surveillance podcast

- Key - REASONABLE AND PROPORTIONATE
- 3 major acts
 - Data disruptions: Law enforcement can apply for orders which stop crimes from taking place by changing data.
 - Adding copying, deleting copying data held in a computer
 - Network activity warrants: access to data
 - Assist in collecting intelligence relating to criminal networks
 - Account take over – can be issued by magistrate
 - If crimes are suspected of being or having been committed. They can apply to take over to account to collect evidence
- Government has power to access data, telecommunications
- TOLA Act (Telecommunications and Other legislation Amendment) 2018
 - Changed previous provisions
 - Inserted part 15 into telecommunications act which attempted to solve going dark
 - Designated communications provider (deliberately broad) e.g. whatsapp, meta operating in Australia needed to abide by similar rules as telecom providers (telstra, optus etc)
 - TAR technical assistance request

- Voluntary request that the communications provider may choose to comply with, then the provider is not liable for things done in that request (if they need to breach company customer contract)
 - Protects providers from liability
- TAN technical assistance notice
 - Mandatory
 - Approved by senior agency and request consultation
 - Providers need to provide technical assistance to agencies where possible
- TCR technical capability notice
 - Mandatory
 - Approval by attorney general and a minister (double lock approval within cabinet)
 - Used when agencies need to ensure that designated communications providers are capable of providing assistance
 - Harder to get than a TAN (requires more assessors e.g. technical SMEs)
- Doesn't bypass warrants process to obtain data. Mostly ensures that data is accessed in plaintext once a warrant is issued
 - What can people/companies asked/required to do
- Needs to be reasonable and proportionate but also technically feasible
- Innocent third parties caught up in surveillance also need to be considered
- When these are issued the oversight agency must also be notified so that they are aware that these powers are being enacted and for what duration
- Systemic weakness/vulnerability - Safety monitor said they are the same (Richard agrees)
 - Vulnerability can be exploited but weakness are vulnerabilities where the exploitation is not yet known
 - Agency cannot ask providers to do things which expose them to system weaknesses
- We don't know for certain what goes on: Secrecy of agencies
 - Hypothetical: Suppose whatsapp is being used by a person. Whatsapp has encrypted data. Agencies can go to whatsapp asking that in their next app updates for this person serve them a fake update

that changes their messages to plaintext so that we can then use this to gather evidence.

- Does this impact one person? Or things more broadly?
 - On surface looks like it only impacts one person. One way it might impact other people is if this becomes known that whatsapp allows agencies to introduce fake updates that they will stop updating their own device and idea flows on exposing greater vulnerabilities to others
- If the government can do this, its a weakness
 - Whats stopping the bad guys from doing the same thing? Real world has symmetry. Should act in accordance with anything you can do, they can do too
 - Updates have historically been the way to break into systems
- How do we as the public decide whether agencies are acting within their limits?

Week 7 Tuesday Notes:

- Data breach simulation
- Data and big data

Data and big data:

Data Governance Plans:

- Someone in charge of data
- And someone in charge of privacy

Someone in charge of data:

Data can be valuable or valuable one day! Which is why, people still keep old data 😞

Moral of the story: Data is valuable.

In Cybersecurity, data is toxic and we should delete it! Hackers won't be able to get our data if they hack into our system if we deleted the data.

How data can be used against you:

- Doxxing
- Blackmail
- Identify theft
- Secrets of you revealed
- Authentication attacks

After hearing Richard's story that is why you don't leave your data out in the open. Even one data or one minute data leaked out can be a breadcrumb to other data about yourself (or even important data about yourself).

Even with data being breached it can lead to getting people who are close to those people whose data being breached have their contacts leaked as well.

Lady lecture's podcast:

What data might be collected?

What can the government do based on the laws?

It seems that mobile phones are being tracked in Australia.

Car surveillance, cars are being tracked through signals sent from tyres, Tesla is already doing it. But Bluetooth and the RFI signals are weak (7m in radius), so a lot of cars on the road make it messy to identify certain cars to track down.

Summary: Think about the hard questions before the crisis hits! For example, which external experts will we call if we run into a data breach. Think about it well in advance, so you get the best person rather than a person you randomly met recently. This leads to you picking the most available person rather than the best person. You need to think about every question in advance.

Movie Tonight:

Data Breach Simulation (Kahoot)

Scenario 1: Suspected cyber event initial discovery

Call external experts for legal, forensic, and crisis advice

Ask IT to investigate and recommend next steps

Tell Jason to leave the chatroom and stop discussing the issue externally

Notify employees via sms/email to avoid logging in or accessing network

- Depending on maturity of the company, call external experts and communicate with IT team
 - IT team might not be well versed or very experienced in this scenario
 - If big company, try to handle internally, if small and don't have a well resourced team, consult external experts
- Internal people should be trained BEFORE crisis happens
 - Should know NOT to publicise the info
 - Jason should be fired'
 - No one really checks their emails. Its easy to say something, much harder to hear something.

tldr; all are good!

Don't you wish you thought about this in advance? What would you do if you had a time machine?

Scenario 2: Communicate about ransom

Jason's brother in law (FBI) to manage negotiations

Previous legal counsel for legal and strategic guidance

CISO's IT forensics vendor whom she met at a recent conference

Notify the insurer and ask the broker for referrals

Generally hackers would exfiltrate the data then threaten to leak the data and/or encrypt the current database so that it would be unusable by the company. Leaking is a big financial risk.

'Red Tape': Companies hate it, but it's good for security! ie. more regulations (data breach notification)

- Jason's brother in law: NO.

- Previous legal counsel: Everything you say can cause you a class action, YES straight away.

- IT forensics vendor: Not at the point where we know who's good and who's not yet, if the CISO trusts then maybe okay

- Notify Insurer: In insurer's interest that the disaster is minimised (will help you find good resources), YES

Scenario 3: What do you do for business as usual? There is SUS activity and some encrypted client data is critical.

Should the committee notify any third parties now?

Yes, notify all major clients, as their medical data is likely impacted

Yes, notify the OAIC since this is a suspected notifiable data breach

Yes, notify law enforcement immediately

No, wait for legal and forensic advice before notifying anyone

No, seek advice and consent from insurer before deciding

- Good idea to seek legal and forensic advice FIRST

- Need to tell clients SOME stuff, should consult legal first before doing something that you can't take back

- Wait a bit before OAIC, risky and you have a period of time

- Law enforcement may have a coordinator that can help

Scenario 4: Will you pay the ransom?

- Depends on the context
- Risks: The threat model, what are you worried about actually happening? what is the problems with the data brach? What assets need protecting?
- Losing access to data (business continuity)
- Trust to stakeholders and customers (reputation damage)
- Future liability (financial penalty)
- Customers
- If pay ransom:
 - have business continuity, protect reputation
 - but data may not be deleted by hackers, may still have copies
- Most companies pay the ransom!
- (You can potentially get in trouble for paying, need to report when you pay off ransom)
- Hard to keep a secret, someone in company will leak

Week 7 -Thursday: Pentesting

Only do this if you have consent!!

Kris' pentesting methodology:

1. Recon – find where things are vulnerable
2. Planning - plan through your process, think about what you will do after the exploit succeeds
3. Exploit
4. Persistence – how do you maintain that access point to the network?
5. Reporting – responsibly disclose all vulnerabilities found

Recon

Nmap – port scanner; displays all open ports and their type

Certain ports are commonly used for certain services – 22 for SSH, 80 for HTTP

Changing ports is security thru obscurity

You can avoid many of these vulnerabilities by restricting permissions appropriately – each user should only have permissions for the things you want to allow them to do

Cap HTB Walkthrough

User flag

- - Download and open tcp packets with wireshark
- - Noticed that the pcap files started with index 1, 2, 3 etc. everytime the scanning was run
- - Set the URL to fetch index 0 (what nathan was doing before we logged in)
- - Download 0.pcap and open with wireshark
- - Inside there is a username and password (unsecure packet so you can open)
- - Try the password to SSH into machine (guess that people like to use the same password for everything)
- - cat user.txt

Root flag

- - Host a webserver as the user account and send file for linPEAS program
- - Chmod and execute the program to scan for any vulnerable points on the computer
- - Noticed that you can use python to set uid on the machine
- - Type python3 in the terminal and try to set our uid to root:

```
```
```

```
import os
```

```
os.setuid(0)
```

```
```
```

- Doesn't quite work because it only works in the python subshell, so instead use python to start a bash shell, and then you can just cat root.txt to get the flag

Week 8 – Monday

Properties of cryptographic hash functions (subset of hash functions)

What properties makes a hash good?

Size

- - Longer hash means more unique inputs -> unique outputs
- - ie. better collision resistance
- - But slower to compute and harder to store

Confusion

- - Make relationship between plaintext and hash harder to identify
- - Don't want there to be patterns that can be seen when inputs modified
- - Harder to reverse the hash

Diffusion

- - Aims to change about half of the output bits every time the input is changed

Suppose $\text{hash}(m)$ is some cryptographic hash function

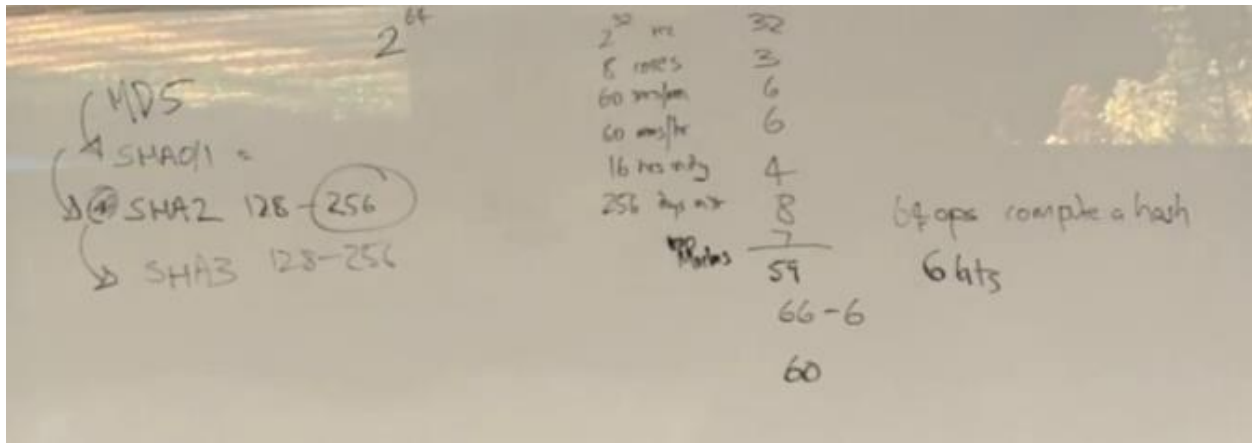
1. **Collision resistance** (computationally infeasible to find $\text{hash}(m_1) = \text{hash}(m_2)$ where m_1 is distinct to m_2)
2. **Pre-image resistance** (given the hash digest h , it is computationally infeasible to find the input m_1 such that $\text{hash}(m_1) = h$)

3. **Second pre-image resistance** (given a message m_1 it is computationally infeasible to find a second distinct message m_2 such that $\text{hash}(m_1) = \text{hash}(m_2)$)

Given those properties we can use cryptographic hash functions as building blocks (primitives) for more complex systems.

Example

Prior to election claim to know who will win. Hash the (name + salt) then verify after the election.



Think about the birthday game:

- - the more birthdays are declared; the more chance of collision is going to increase.
- - The approximate number of birthdays you go through is approximately $\sqrt{\text{total number}}$. I.e. total possible number of birthdays = 365. Approximate collision would have been ~23.
- - This applies to hashes too. If there is a 128-bit long hash, to find one, it would take 2^{64} tries.
- - Exam question?!?!: I make a will, and I leave everything money to one of my sisters. I do not trust my sisters, so I hash it, and I send the hash to the lawyer, and place it everywhere. When I die, everyone can then check the integrity.
 - However, the lawyer can be a little fiddly. So, I place salt into the will at the end, so people cannot brute force it. But the lawyer, before they hash it, writes an identical document, and hash both. And if they change it little by little, they just need to work out 2^{64} fiddles to make it collide.
 - Check out the calculation

The most common hash right now is SHA2. There are still weaknesses, but SHA2 is better than MD5.

How do we hash a very long message?

- We can break it into chunks
- There are few block modes:
 - Electronic Code Book (ECB): break them into chunks, encrypt each chunk and concatenate each chunk together.
 - Cipher Block chaining (CBC): Take the plaintext and divide it into chunks. Start off with an Initialisation vector and take an XOR onto the encryption to obtain a jumble, and then apply the encryption (same length). Here is the interesting bit: for the next block, we use the previous ciphertext to XOR the plaintext to jumble it more, then encrypted. And we keep going. Its also faster to decrypt.
 - Counter (Richard's favourite): you can do all of them in parallel. We use nonces: Arbitrary values that we use only once. We apply nonce and a counter (counter is a counter, which blocks is which) with the encryption, and then XOR it with the plaintext to get the ciphertext. The nonce stays the same.

Basically, we use a washing machine algorithm.

Simpson desert with the telegram message:

- Mitigation: I send the message, and compute the hash, but have a salt (such as sausage). I send the hash with sausage at the end. $H(M|sausage)$. They do not know the password which I appended it, so they cannot compute a valid hash. I then send the message with the hash (with the sausage thingy). This is called a MAC (message authentication code), and this makes it tamper evident. Here it is not a salt - it is publicly known to make it hard to have collision. Here, it is a secret. This does have a weakness (if we put it at the beginning). A bad guy can then change the message and update the MAC - a second preimage attack. This is vulnerable to a length extension attack! This relies on knowing the public algorithm, XOR, and how it works, and how the algorithm folds onto itself. What weakness does it have if we put it at the end?
- Modern day we use a H-MAC.

Please correct this if it is wrong.

Week 8 – Tuesday: Errors/Safety/Security

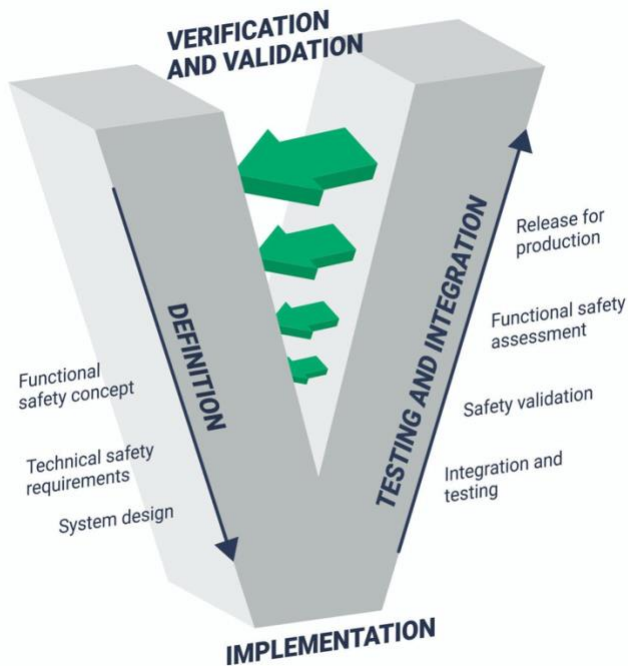
How to make a system safe? Is it the same with security?

Safety v. Security:

- Safety:
 - up against randomness, incidents?
 - Usually it is used for “people”
 - NIST-diagram, pen-test security, using probability to somehow calculate the rate of incidents happening.
 - 1 in 10^{10} that someone may die (?)
- Security
 - up against an adversary, someone deliberately trying to attack, that is the worst case scenario that might happen.
 - It may be used for “people” or even a lot of other things, such as: assets.
 - We can’t use probability to account for the rate of an attack happening.

So the question is what can humans do in design and operation that they should not fail?

V-MODEL: Definition – Implementation – Testing and Integration + Verification and Validation



Herald of Free Enterprise 1987 Accident

- Ferry for cars in Belgium that departed with the loading doors open resulting in it flooding itself
- Cause, the person responsible for making sure the doors were closed had gone to sleep before departure, the second person that was meant to check everything was alright had assumed everyone had done their jobs and finally the third person off the boat deck had seen the doors were open and had not reported it since it wasn't their job when asked why.
- This shows how people are living in rule based systems, focusing on only their responsibilities and not others
- This begs the questions when people should move between the different levels. It is this something that can be solved with my skills, if not with the rules, if not is it time to my knowledge
- Cultures and rules: If the culture is focused on obedience to rules and you were training people to respond to incidents, you face a challenge in

making sure people can change their behaviours even in the context of their society and culture to flip the switch from rule-based to knowledge-based decisions or actions

- **This is helped by feeling relaxed comfortable and hindered by stress. So it is only by creating a culture where people feel supported safe and at their best that people are able to make wise decisions and move up the levels**

Week 8 – Thursday (Red Teaming vs Blue Teaming)

How are we gonna defend ShatGPT? (Blue Teaming)

<https://shatgpt.securitygrounds.org>

How did they trick openai to give the keys though to bypass SQLi???

Answer: User sends input to the AWS Cloud (Chat Web App (Vue)) which then sends the input to the AWS API Gateway + Lamda (Lamda function), the Lamda function then sends the input to the OpenAI for it to be summarized. The summarized input is then sent back to the Lambda function, Lamda function then asks the database it has for results (all important, the rest are now non-important) and the results is sent back to the AWS Cloud and it sends back to the user. When u are asking the OpenAI, u asking the SQLi injection to be in the payload. (Last sentence is exactly what you are trying to achieve)

WAF rules: Web Application Firewalls that you put in place to allow, block and monitor web requests based on them satisfying certain conditions

- Can be a default; good at detecting generic attacks

Requirements on how to defend your web application

User-Stories: Define requirements for each user/customer. How do we build it and how to get there.

Requirements:

- Acceptance criteria: What do I need for it to be satisfied
- Estimation: How much time/effort it will cost (ideally not a long-term thing)
- Technical specs: More developer focused details

(By Johnny Qin)

What is threat modelling?

Threat modelling is on a whiteboard, identify what threats there are; what can go wrong

- Primarily come up with solutions for how to solve them

Infrastructure As Code (IAC)

Ability to manage automatic deployment and maintenance of code, using code, rather than through doing stuff manually (like dogwater AWS console) (Terraform is an example of this)

OpenVAS, Burpsuite, Bandit, Semgrep

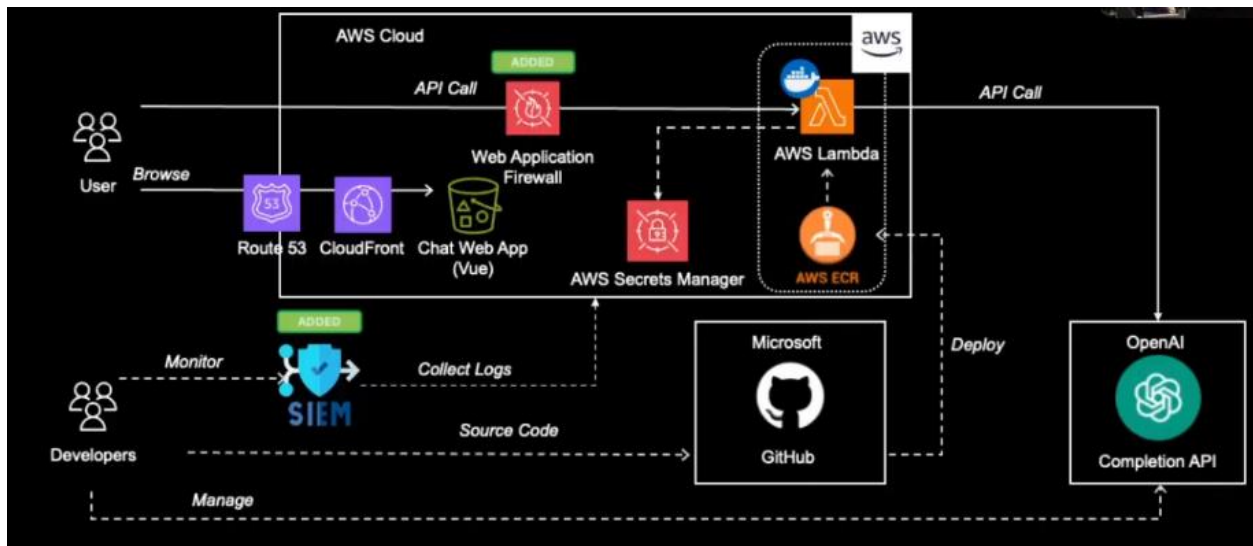
Triage: To do a preliminary assessment of the dangers

SIEM: Security Information and Event Management

- Solution that helps organisations detect, analyse and respond to security threats; usually in the form of a dashboard
- E.g. Splunk, Elastic

Risk-Based alerting

- Risk objects (Users, Endpoints)
- Risk scores (Arbitrary values given to events/other detections)
- Risk rules (Risk score exceeded for risk object in last 24 hours)



Week 9 – Monday

Chess Puzzle

Book – The Chess Mysteries of Sherlock Holmes

Problem: How can you prove that at some point in a game, a pawn was promoted?

- Two white bishops on same color square

TLS / HTTPS / SSL

In the old days, emails and information sent across the web was open for everybody to see

- Not confidential
- None or limited encryption
- Had to assume no one else was sniffing
- One solution: Encrypt emails before sending and decrypt upon receiving

How do we send information between each other confidentially?

- Can you use a sequence of bits to encrypt information?

[One-time pad - Wikipedia:](#)

Given the plaintext which is a series of bits, and you have a key of the same size, you can XOR it or add it to the bit sequence to get encrypted text. Then you can decrypt it using the same key. One time use makes it extremely strong.

One-time pad

| | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

| | | | | | | | | |
|-----------|---------|----|---|---|----|---|----|-----------------|
| Plaintext | T | 19 | + | F | 5 | = | 24 | Ciphertext YZWU |
| | E | 4 | + | V | 21 | = | 25 | |
| | S | 18 | + | E | 4 | = | 22 | |
| | T | 19 | + | B | 1 | = | 20 | |
| | Keyword | | | | | | | |

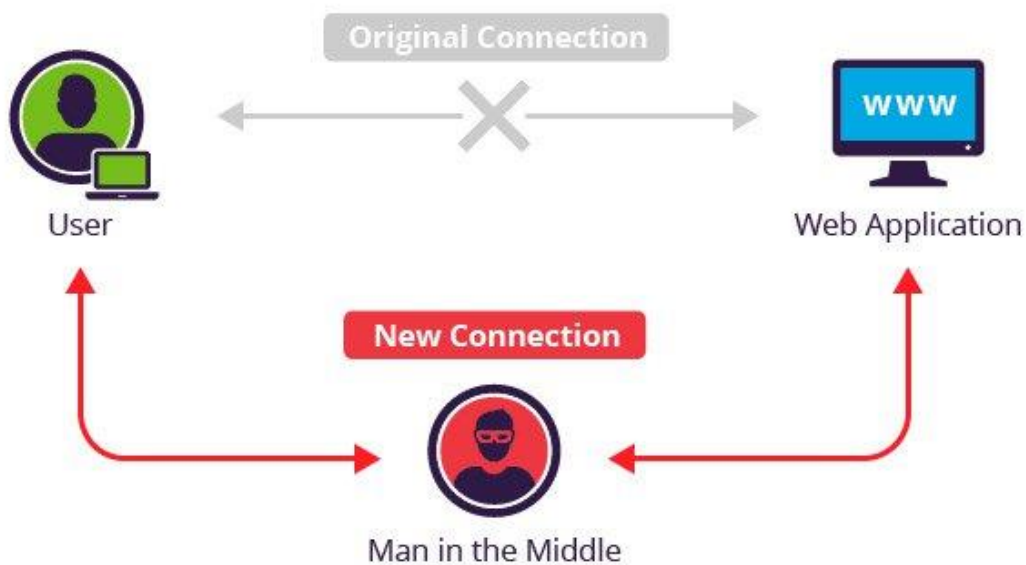
Source: Andrew Farnham
Used throughout all rights reserved

Almost impossible to brute force, the amount of entropy in the key is the amount of entropy in the message.

How the lack of carbon paper supply affected the strength of German secret codes:

[Cipher Department of the High Command of the Wehrmacht - Wikipedia](#)

Man in the middle attacks:



How do we find public keys?

- Wikipedia page,
- Google

But how do we know the public key is real? An attacker could publish a fake public key (Fundamental problem of security).

PKI

People usually go straight to mitigation out from their minds, and diverged away from what are the most precious assets they should protect, famous example 911 -> national security, great example, running to a drug addict that's trying to get into your car.

Assets -> risks around the assets -> mitigation

Houdini case study, burying a physical thing and let the society know what he buries, and convinced on he really buries something.

Week 9 –Tuesday

- Openness

- Transparency, honesty, willingness to try
- Teamwork
 - Work together to achieve something than an individual cannot achieve
 - Not trying to keep the glory to yourself
 - Acting to the benefit of the whole team
- Communication
 - Communication is about the person you are talking to, not yourself
 - To communicate effectively --> understand your audience

Week 10 Monday

(I only came around 6:30pm, please if anyone was around from 6pm to 6:30pm, please fill in what Richard said here)

“NASA in the end did not learn their lesson and they screwed up another time again. Safety culture was STILL NOT THERE after the first time they screwed up” – Richard Buckland

Organizations need to change their mindset and priorities in order to adhere to the safety culture.

The Challenger launch decision, great book by Richard Buckland

When a crisis happens, immediately switch to high gear and prioritize the crisis, think about it to think of solutions to prevent the crisis. Stop everything at once and FOCUS on the crisis! Be in the frame of prioritizing the crisis. Have a frameshift when a crisis happens. Unfortunately for us humans, it's not very easy for us to immediately have a frameshift when a crisis happens 😞 I sometimes wonder how we as a species survived even during the caveman times??

Frameshift Frameshift Frameshift

Btw, chess puzzle was solved 😊 Yay!!

Command & Control

Command & Control is the system that an order is actually carried out and executed. One type of Command & Control system is the Pyramid structure.

Pyramid Command & Control system/structure

Chain of command needs to be followed. U can jump the chain but it's like a bad form. Essentially u can can jump the chain but it depends on the command & control.

Cons:

- Single point of failure (for e.g. the top of US army is the US president)
- Examples of the first point: Leader becomes arrogant and corrupt
- Example of the first point: Leader becomes entitled to greater things when they actually do not

Pros:

- More coherence (all attack at once)
- Less latency

The below examples are pyramid command & control:

- Go go power rangers!!! XD (The power rangers all join together to form that giant robot)
- The guy in the machine in the Avatar movie (blue people)

Checks & Balances command & control (opposite from the pyramid structure) examples:

- Greek kings giving themselves more wealth, people have to suck up to them so they decide to overthrow the king.
- People realise they needed new king: they had a fear where the new king becomes corrupt & arrogant.
- Another example, communists overthrowing the Tsar in Russia --> Stalin the dictator ruling Russia
- Democracy is an example (Richard says democracy is not really an example of Checks & Balances command & control but some countries that have democracy have lots of checks & balances (UK, Australia, Canada, India, US etc.) while other democratic countries don't have many checks & balances (Russia)

Pros:

- Limits unequal distribution of power.

Cons:

- Person with limited power will try remove the checks and balances to gain more.

Examples of checks and balances:

- Leaders can only be leaders for a certain duration (for example in the US it's 4 years I think) 2 terms of 4 years, so 8 years
- Leaders are split into leaders of different sectors
- Dual control (ice cream van with two teenagers example)
- Double entry bookkeeping (requires corrupting two ledgers)

White van story:

People claim that their boss mistakenly brought more than enough speakers and their boss does not know it happened. so, they have stolen the speakers from their boss and now want to sell them. do you want to profit from their bosses foolishness. And so these people want to sell it to you. They claim they are really good speakers where in reality they aren't really good speakers. And you pay them a handsome amount of money for those speakers thinking you are getting a bargain and tricking their boss whereas they are really good without realizing that you just got ripped off cause u paid a lot for speakers which aren't really good at all.

Puts victim in "hot" state through time pressure.

Invokes loss aversion, they don't want to miss the good deal.

Invoke a sense of tricking a boss and being a bit outside the law.

Cyberwar

Computer security flaws are agnostic to money. They are just a product of complex systems.

1st domain of war is land

2nd domain of war is sea

3rd domain of war is air

4th is space

5th is cyber - Cyberspace

Russian hackers used DDOS to launch cyberwar on Estonia. North Korea has also joined in using cyber to attack their enemies. Sadly, it is government sponsored cyber-attack :(

Nations now have “cyber command”

Kinetic impact is when a cyber-attack has an impact in the physical world.

How would we target military cyberattack in Australia? ideas people had:

- Electrical routes of Sydney trains (target them first)
- Target those infrastructure that are connected to the internet first
- Reroute planes into buildings?
- Attack an airport? (They need to be connected to other airports to have flight data?)
- Stop internet from working
- Stop powerplants
- Attack shipping systems
- Attack government websites
- Attack stock market
- Attack election
- Take out water filtration systems
- Attack identity systems, driver licenses
- Information warfare (People getting wrong info from the internet)

Week 10 Tuesday

Admin:

Revision session next Tuesday same time as usual lecture

The Secret Ballot voting system is very useful in making sure there is no vote buying and reduces the fear of intimidation. However, the main problem is that the system is not

visible. We do not know if it has been tampered with, if the right people have voted, if people have voted more than once etc. If we store partial information about people, there is a risk that they could be identified.

Knowing yourself, other people holding you accountable

(please write notes for the discussion about the movie!!)

Bad authentication – Mandrake only executes the order because he recognizes General Ripper's voice

Voting and Election Integrity:

1. Influence of Elections: Campaigns can manipulate voter turnout and demographics.
2. Control by Incumbents: Incumbents may influence elections by appointing biased officials.
3. The Secret Ballot: Secret ballots protect voter privacy and ensure unbiased voting.
4. Problems with Electronic Voting: Electronic voting is vulnerable to errors and manipulation, and complex cryptography hinders trust.
5. Past Failures in Electronic Voting: Software bugs have led to incorrect vote recordings in past examples.
6. Blockchain and Voting: Blockchain could secure elections but faces complexity and trust challenges.

Exam Strategy for Security Course:

1. Read and Follow Instructions Carefully: Thoroughly understand questions and follow formatting to avoid losing points.
2. Answer Concisely: Short, clear answers are more effective than lengthy ones.
3. Problem-Solving Mindset: Focus on understanding problems before jumping to solutions; human factors are critical in security.
4. Command and Control in Complex Systems: Learn from case studies like "Dr. Strangelove" to understand high-stakes security environments.

Cybersecurity Insights:

1. Self-Awareness and Leadership: Understand emotional triggers and leadership styles in high-stress environments.
2. Mental Health: Address the emotional toll of cybersecurity work for better long-term effectiveness.
3. Cybersecurity as a Team Effort: Success relies on collaboration and a well-rounded team.
4. Emerging Technologies: Stay updated on AI, quantum computing, and other disruptive technologies.
5. Strategic Thinking: Cybersecurity is a dynamic game requiring adaptability and constant learning.
6. Career Opportunities: Cybersecurity offers continuous growth, with diverse career paths for professionals.

Dr. Strangelove discussion

what strikes you from a security point of view?

- single point failure (multiple single points of failure)
- human error causing an insider attack
 - personal incentive/COI (Conflict of Interest)
- redundancy
- lots of authorisation checks, but ultimately was useless
- overall protocol looks good but had single point of failure in actuality
- 'failsafe point' - planes were started at failsafe point
 - should we autolock protocol to keep going after a certain point? (flight plan R having no turn back point)
- command and control structure
- project R should have only run if president or capital was compromised but wasn't actually the case
- man in the middle
 - russian ambassador speaking in russian, trusted position but could have mistranslate
 - secretary had to relay the information to the general which could have been a point of failure
- project sundial
- game theory about mutually assured destruction
 - if they attack us the response will be so terrible to deter people from attacking
 - doomsday machine in movie
 - the missile gap (US got scared because of russian misinformation)
- sunk cost fallacy: already sent one bomber plane why not go all in
- chain of failure: missile broke the code machine, which resulted in pilots not receiving callback code
- military authenticated by uniform
- side channels to discover the fallback code
 - Secret leaked even after death
 - Best number of people to know a secret is 0
- type 1 type 2 errors
 - false positive: launch second strike w/o first strike
 - false neg: not launch second strike when first strike landed
- social engineering between US and russian pres to prevent overreaction

- hierarchy of power steep (overriding commands)
 - to what extent can people below question orders
- failsafe
- human to human interaction - calls up the russian president even through there is a bad situation
- dr strangelove: got super psyched about computing remaining humans with the computer, but exempted themselves
 - politicians always look after themselves first
 - ethics is hard
- comedy thing; there was never a joke made about the president, always targeted towards the generals, nazi
 - emphasise seriousness and actuality of people the filmmaker wanted audience to support

Key Characters (if you forget or need spelling):

- General Jack D Ripper – General who has gone crazy and calls the attack
- Group Captain Lionel Mandrake – Captain who is trapped with Ripper
- President Merkin Muffley – president of the USA
- General Turgidson - Chairman of the Joint Chiefs of Staff
- Alexei de Sadeski - the Soviet ambassador
- Premier Dmitri Kissov – leader of the USSR
- Major T. J. "King" Kong – the Major on board the plane which drops the nuclear bomb
- Dr Strangelove – Doctor who informs the president of what the Doomsday device is
- Colonel “Bat” Guano - Colonel who finds Mandrake

Key Scenes:

3:22 – Mandrake receives the call from Ripper

- do you recognise my voice? Issue: no other authentication done. Could be a recording or impersonation

19:25 – Mandrake goes into Ripper’s office with radio

20:42 – Mandrake realised Ripper has gone mad and tries to call off the attack (he’s locked in the room and cannot communicate with the outside)

25:00 – Discussion about why plan R exists (if the chain of command was disrupted a general could issue a nuclear attack)

37:00 – US soldiers fight US soldiers

47:00 – Doomsday device 49:00 Doctor's explanation

57:11 – Mandrake's last attempt to get the code from Ripper (before he kills himself looking like he shaved)

1:02:38 – Peace on earth

1:11:30 – Calculating fuel (discussion with president and premier after about low flying planes)

1:15:17 – recalculated fuel (change target)

1:23:09 – Nuclear bomb + Dr Strangelove idea to keep people underground