

Systems Security

COMSM1500

Announcements

bristol.ac.uk



Bristol Ethical Hackers

- A weekly meeting where we will learn how to hack.
- We will also
 - Enter capture-the-flag competitions
 - Invite industry speakers
 - Find vulnerabilities in devices
- Sessions are Wednesdays 3-5pm
 - For the next two weeks in MVB 1.11a
- Contact Joe.gardiner@bristol.ac.uk for questions
- Join our Facebook group (“Bristol Ethical Hackers”) or Discord Server (<https://discord.gg/HJGpzUm>) for announcements. We are also on Twitter @BristolHackers

Fully Funded PhD position

- £22k a year (tax free) + fees
- £8k a year research budget
- British National Only (funded by the GCHQ)
- Work on intrusion detection
- At the intersection of
 - Machine Learning
 - Systems
 - Human Factors
- <https://www.jobs.ac.uk/job/BVQ074/phd-studentship-towards-interpretable-and-actionable-provenance-based-intrusion-reports>



Today Lecture

- Safety VS Security
- Security Vocabulary
- Policy, Threat Model, Mechanism
- Spectre/Meltdown family of attack (a brief intro to...)

Safety



Safety terminology


- Accident: unplanned and undesired event that leads to a loss
- Hazard: condition that can lead to an accident in a particular environment
- Incident: sometimes, a hazard that does not lead to an accident
- In security only “security incident”, it does not “happen by accident”

Safety and Security

- Safety: freedom from accident (caused without intent)
- Security: freedom from incident (caused maliciously)

Safety measure

1. eliminate hazard
2. reduce likelihood of occurrence
3. control
4. mitigate (reduce damage in case of accident)



Most effective

Active/Passive

- Passive safety measure: presence alone contributes to safety
 - e.g. fences
- Fail-safe: fail in such a way that the system end-up in a safer state
 - e.g. fuse
- Active safety measure: monitor the system and intervene
 - e.g. fire tower (forest fire)

Risk

- $\text{risk} = \text{loss} * \text{probability of occurrence}$
 - Favored: reduce probability of accident (to zero if possible)
 - Reduce loss
- In most case it should not be acceptable to not reduce occurrence of hazard if possible
- Safety industry ALARP (as low as reasonably practical)
 - Must demonstrate further reduction is either impractical...
 - ... or unreasonably costly

Security



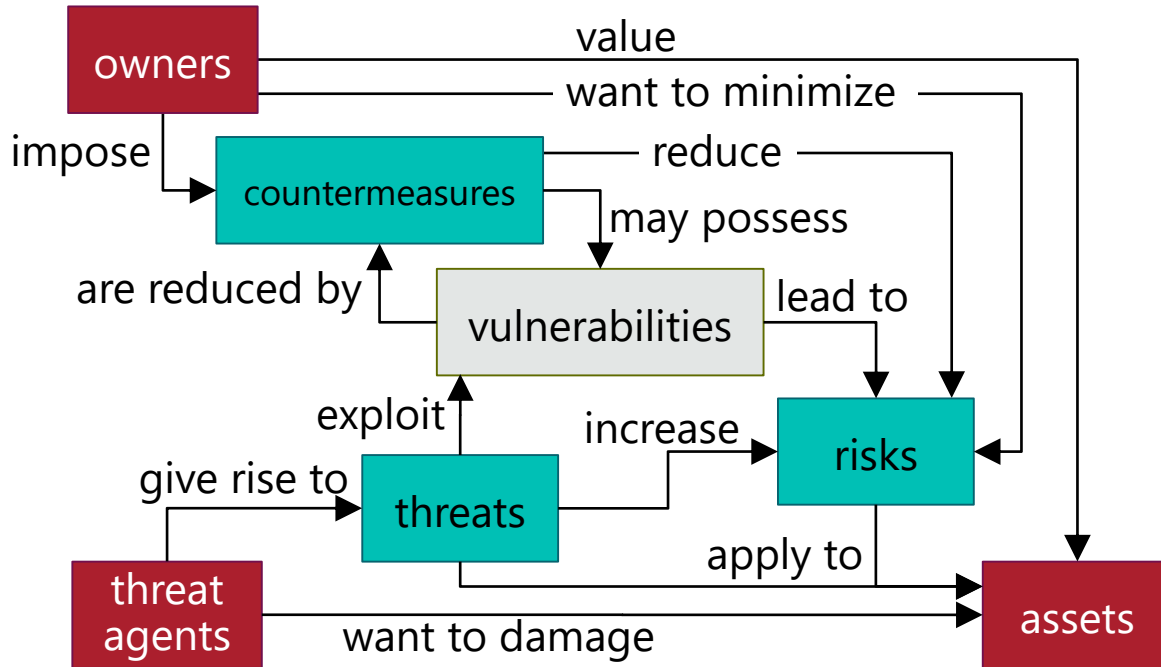
Computer Security

- *Protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability and confidentiality of information system resources (includes hardware, software, firmware, information/data and telecommunications).*
 - US National Institute of Standards and Technology 1995

terminology

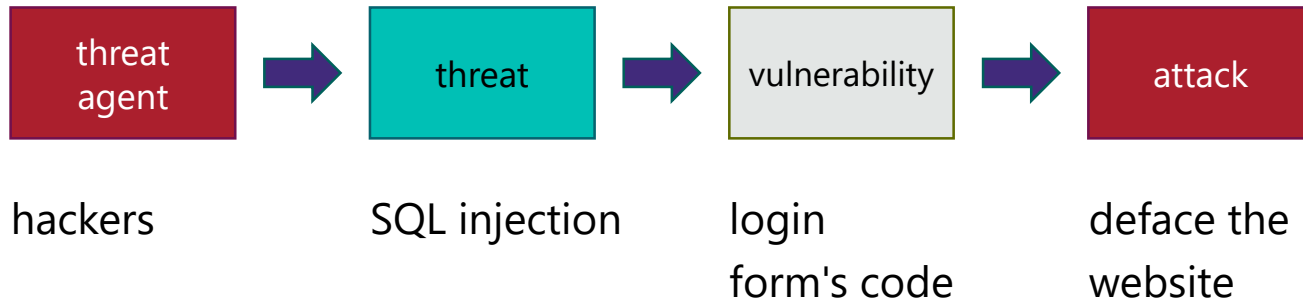
safety	security
hazard	vulnerability
(no equivalent)	threat
safety measure	countermeasure
risk	risk
accident / incident	incident

Terminology (IETF RFC 2828)



attack

- realisation of a threat
- sequence of actions that results in a loss



countermeasures

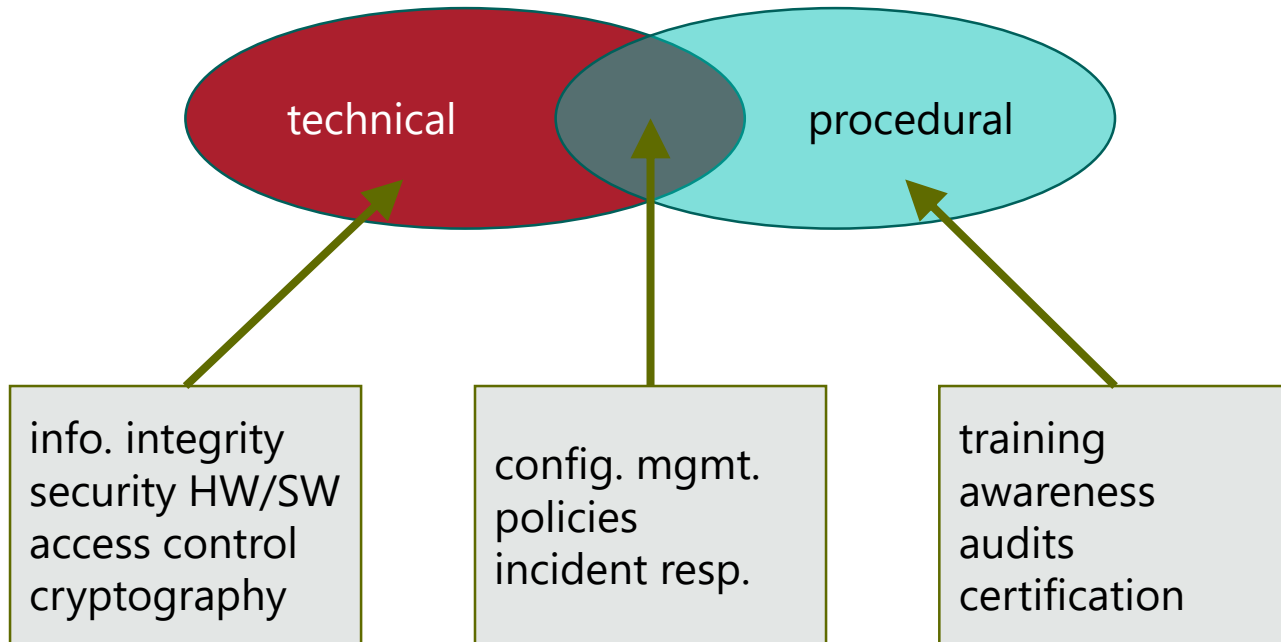


prevent

detect

recover

countermeasures



Risk (security economics side notes)

- $\text{risk} = \text{loss} * \text{probability of occurrence}$
- Think about CC fraud
 - Cost of preventive measure (i.e. reduce probability)
VS
Paying for damage
 - Bank prefer to pay for damage (simple cost calculation)
 - Simple risk calculation
 - $\text{Risk} < \text{cost of preventive measure}$
- In other industry:
 - Data leaks, maybe some damage to your image
 - GDPR fines for personal data leak (up to 4% of annual global turnover)
 - Loss sky rocket
 - Incentive to bring down probability of occurrence

Computer Security

- Policy -> Confidentiality, Integrity, Availability

Computer Security

- Policy -> Confidentiality, Integrity, Availability
 - C: only faculties can see your coursework

Computer Security

- Policy -> Confidentiality, Integrity, Availability
 - C: only faculties can see your coursework
 - I: only faculties can change grades

Computer Security

- Policy -> Confidentiality, Integrity, Availability
 - C: only faculties can see your coursework
 - I: only faculties can change grades
 - A: faculties must be able to enter grades/students to submit their work

Computer Security

- Policy -> Confidentiality, Integrity, Availability
 - C: only faculties can see your coursework
 - I: only faculties can change grades
 - A: faculties must be able to enter grades/students to submit their work
- Threat Model -> assumptions about the adversary, the system, the environment

Computer Security

- Policy -> Confidentiality, Integrity, Availability
 - C: only faculties can see your coursework
 - I: only faculties can change grades
 - A: faculties must be able to enter grades/students to submit their work
- Threat Model -> assumptions about the adversary, the system, the environment
- Mechanism -> Software/Hardware

Why is it hard?

- A faculty can access coursework
 - Easy just need to test!
- Negative goal
 - This is much harder
- Only a faculty can access coursework
 - It is hard need to think about what the adversary may do
 - Guess password?
 - Steal a laptop?
- It is an iterative process (we saw it during the last lecture)

What can go wrong?

- Policy
 - Last lecture
 - e-mail access
 - Amazon example

What can go wrong?

- Threat models

- Human factor! (making assumption about what people would do)
- Wrong assumptions about attacker knowledge (last lecture Apple)
- Change over time (DES)
- Software supply chain poisoning issue in China

**HACK BRIEF: MALWARE SNEAKS
INTO THE CHINESE IOS APP
STORE**



Jennifer Lawrence Phishing

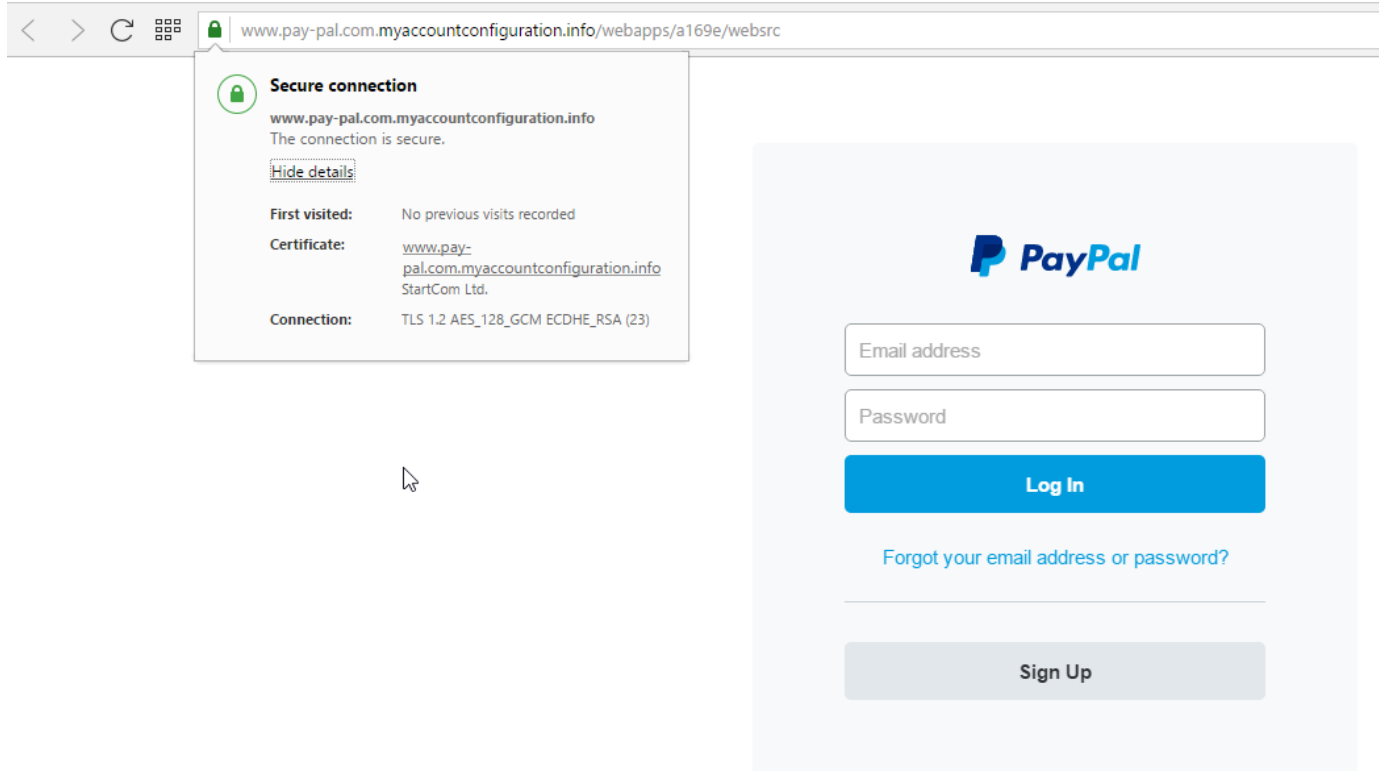
- Simple and easy
- Send e-mail that appears to come from a legitimate party
- Ask for information
- Or link to a fake website looking like the real thing (i.e. please reset your password)

Man behind Jennifer Lawrence nude photo hack sentenced to prison

Edward Majerczyk, 29, of Chicago receives nine-month prison term in plea deal after illegally accessing celebrity accounts and stealing nude images



A legitimate fake site (untrustworthy CA)



What can go wrong?

- Mechanisms
 - That is the source of most vulnerabilities

Example

When mechanisms go wrong



Spectre/Meltdown (made simple)

- Late 2017/ Early 2018
- Lots of chatter online
- Fixes in OS
- Family of exploit to the same underlying vulnerability



Spectre/Meltdown (made simple)

- Late 2017/ Early 2018
- Lots of chatter online
- Fixes in OS
- Computer get much slower ☹️
- Family of exploit to the same underlying vulnerability



Spectre/Meltdown (made simple)

- Problem is much larger
- Every OS is affected
- Modern CPU architecture is the problem
- Intel, AMD, ARM are affected

Spectre/Meltdown (made simple)

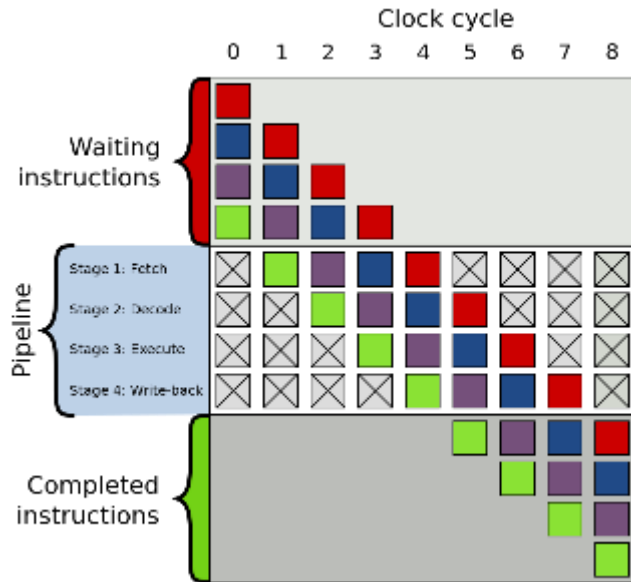
- Problem is much larger
- Every OS is affected
- Modern CPU architecture is the problem
- Intel, AMD, ARM are affected
- Can be exploited by a simple piece of javascript

High risk!

bristol.ac.uk

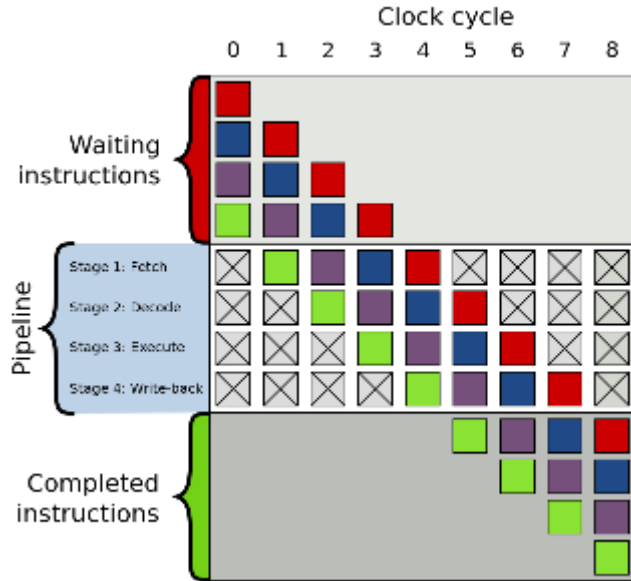


Instructions pipeline



- Trying to keep all part of the CPU busy

Instructions pipeline



- Trying to keep all part of the CPU busy
- Fetch
- Decode
- Execute
- Memory Access
- Register Write Back

Branch and bubble

- Problem when branching
- Evaluate conditions
- Need to wait to get the instructions to put in the pipeline
- This create a “bubble”

Branch and bubble

- Problem when branching
- Evaluate conditions
- Need to wait to get the instructions to put in the pipeline
- This create a “bubble”
- Solution **branch predictor**
 - Make an educated guess of what should be executed
 - If wrong?

Branch and bubble

- Problem when branching
- Evaluate conditions
- Need to wait to get the instructions to put in the pipeline
- This create a “bubble”
- Solution **branch predictor**
 - Make an educated guess of what should be executed
 - If wrong
 - CPU state is reverted (that is good)
 - ... but CPU cache is not (that is bad)

Branch and bubble

- Problem when branching
- Evaluate conditions
- Need to wait to get the instructions to put in the pipeline
- This create a “bubble”
- Solution **branch predictor**
 - Make an educated guest of what should be executed
 - If wrong
 - CPU state is reverted (that is good)
 - ... but CPU cache is not (that is bad)
 - If we guessed wrong we may have (half) executed instructions in the pipeline

Spectre/Meltdown (made simple)

- Exploit branch prediction

Spectre/Meltdown (made simple)

- Exploit branch prediction
- You can get access to data in the cache
- Access check is done when instructions are “executed”
- You can read kernel memory, leak password, keys etc...
 - if (x < array1.size)
 - y = array2[array1[x]]

Spectre/Meltdown (made simple)

- Exploit branch prediction
- You can get access to data in the cache
- Access check is done when instructions are “executed”
- You can read kernel memory, leak password, keys etc...
 - if (x < array1.size)
 - y = array2[array1[x]]
- Check the extra **reading** material
 - <https://github.com/bris-sys-sec/reading>

COURSEWORK GROUPS!

- Groups of 4 students
- (only) 1 e-mail per group
 - thomas.pasquier@bristol.ac.uk
 - Header: [Systems Security] Group
 - Body: id1, id2, id3, id4
- If you cannot form a group you will get one, no need to e-mail me (>80)
- I will circulate the list
- FRIDAY 7PM
- IF NOT IN A GROUP RANDOM ALLOCATION

Thank you

Office MVB 3.26

bristol.ac.uk

