University of Glasgow | School of Computing Science

Networks & Operating Systems Essentials

Dr Angelos Marnerides

<angelos.marnerides@glasgow.ac.uk>

School of Computing Science

Based on slides ©2020 Angelos Marnerides

PRIVACY AND SECURITY (2)

Privacy

- Privacy can have different definitions
  - Legal privacy (e.g., someone's right to be left alone)
- Our focus: ***data privacy*** *(a.k.a information privacy)*
- ***Data privacy focus: use and governance of data***
- *Related standard: ISO27002 (https://www.iso27001security.com/html/27002.html)*

Personal data

Personal data is defined as:

- Any information about a living individual which is capable of identifying that individual.

Sensitive personal data is defined as:

- Any information relating to an individual's racial or ethnic origin, political opinions, religious beliefs, trade union membership, physical or mental health or condition, sexual life, alleged or actual criminal activity and criminal record.
  - Under GDPR sensitive personal data is referred to as "*special categories of personal data*"

General Data Protection Regulation (GDPR)

- EU Regulation (a set of laws) applied in 2018
- UK: Data Protection Act 2018 (inherits from GDPR)
- Data Protection Act: defines your data privacy rights and "*how your personal information is used by organisations, businesses or the government.*"

GDPR (cont.)

- In order to comply with the new law:
  - One must have a legitimate reason for processing data – this covers what and how much processing of the data can be undertaken
  - Consent must be freely and unambiguously given and can be easily withdrawn
  - Data Processing activities must start with “privacy by design and default”

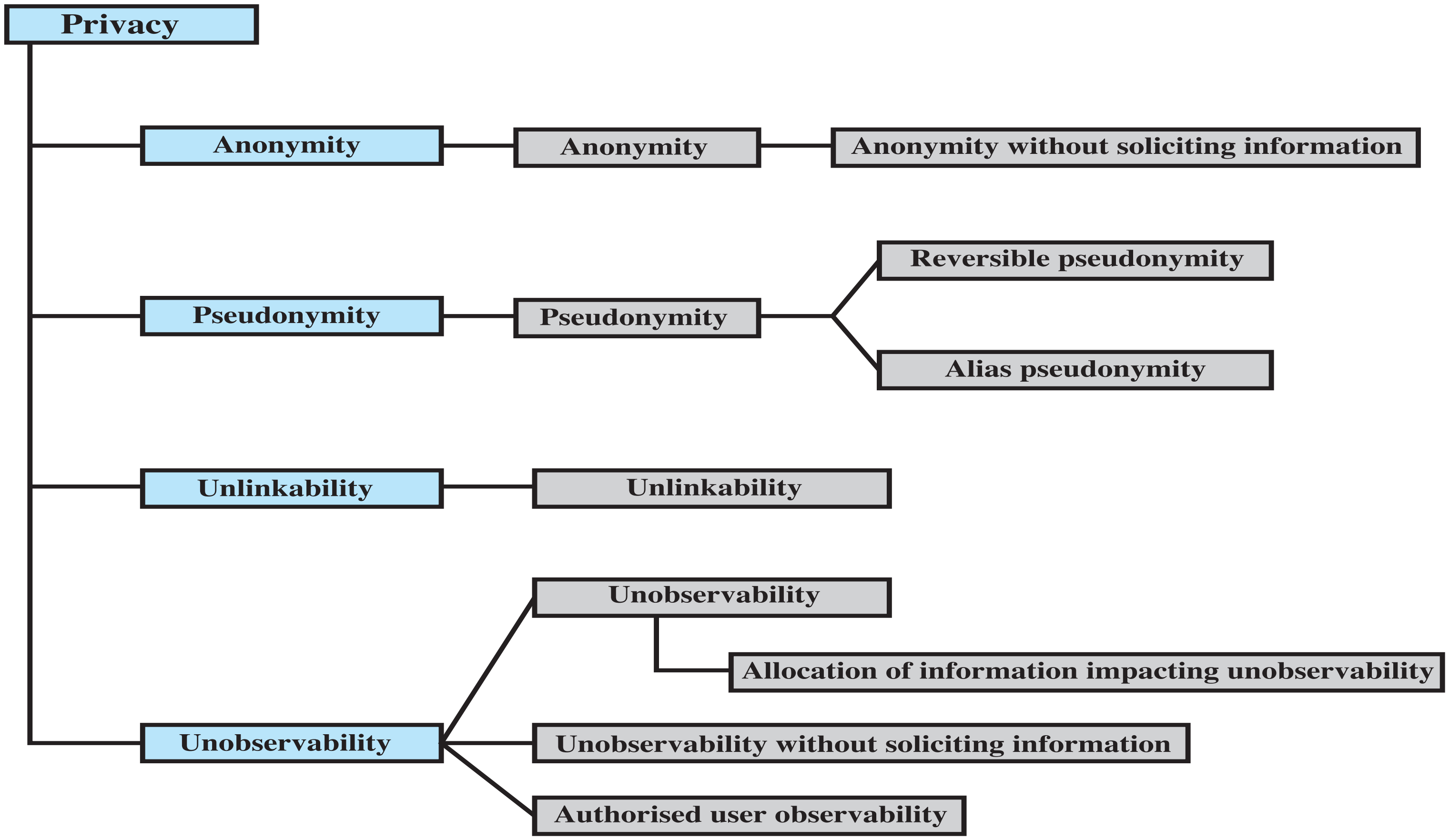
GDPR Principles

- *Lawfulness, fairness and transparency* – as with Data Protection Acts
- *Purpose limitation* – only collect for specific purposes and then do not use for other purposes
- *Data minimisation* – only collect the data needed for a specified purpose
- *Accuracy* – data needs to be correct and kept up-to-date
- *Storage limitation* – data should not be kept for longer than need to fulfil specific purpose
- *Integrity and confidentiality* – data needs to be protected appropriately,
  - E.g. secured through appropriate access control, encryption, etc.
- *Accountability* – a data processor must be able to prove that they are complying with the GDPR regulations

Data Protection Design Principles

- Data Protection requires avoiding harm to individuals by possible misuse or mismanagement of personal data
  - Any data collected, used, or stored about a person therefore is under the auspice of relevant legislation (e.g. GDPR)
- Data protection principles include:
  - Data should only be collect for a specific purpose
  - Collected information can only be used for specific and agreed purposes
  - Records and stored data has to be kept accurate, up to date, safe and secure
  - Information should only be processed lawfully
  - Data subjects should be granted access
    - In line with relevant legislation

Common Criteria for Privacy


Privacy
Anonymity
Anonymity
Anonymity without soliciting information
Pseudonymity
Pseudonymity
Reversible pseudonymity
Alias pseudonymity
Unlinkability
Unlinkability
Unobservability
Unobservability
Allocation of information impacting unobservability
Unobservability without soliciting information
Authorised user observability

Based on slides © 2020 Angelos Marnerides, © 2017 Colin Perkins

DEVELOPING SECURE NETWORK APPLICATIONS

Developing Secure Network Applications

- The Robustness Principle (aka Postel's Law)

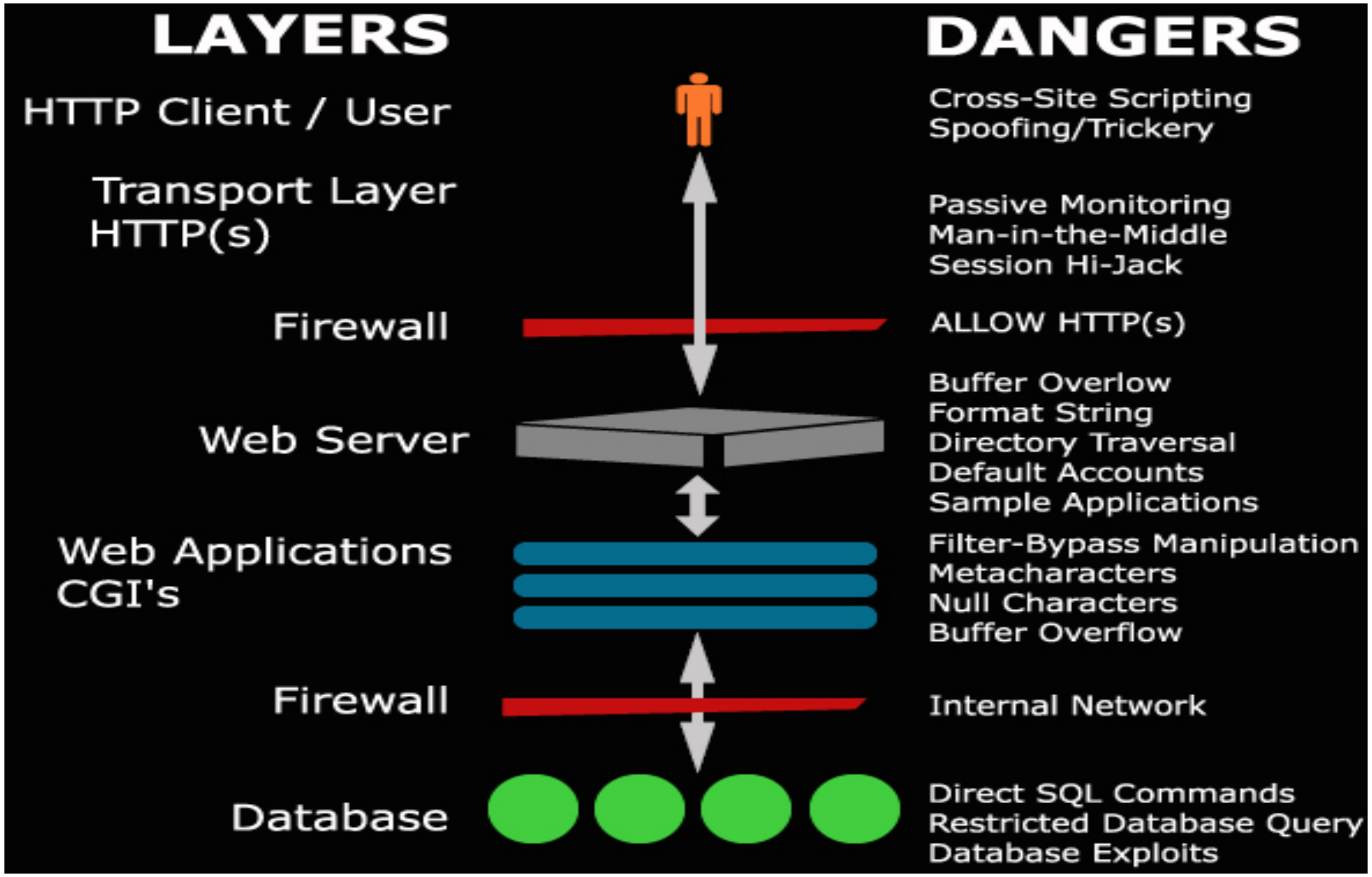
"Be liberal in what you accept, and conservative in what you send"
— Jon Postel, RFC1122

"Be strict when sending and tolerant when receiving. Implementations must follow specifications precisely when sending to the network, and tolerate faulty input from the network.
When in doubt, discard faulty input silently, without returning an error message unless this is required by the specification."
— Jon Postel, IAB RFC1958

"[...] Experience shows that there are negative long-term consequences to interoperability if an implementation applies Postel's advice. [The] flaw in Postel's logic originates from the presumption of immutability of protocol specifications.
[...] Active maintenance of protocols reduces or eliminates the opportunities to apply Postel's guidance"
— Martin Thomson,
https://www.ietf.org/id/draft-iab-protocol-maintenance-00.txt

"Postel lived on a network with all his friends.
We live on a network with all our enemies.
Postel was wrong for today's internet."
— Poul-Henning Kamp

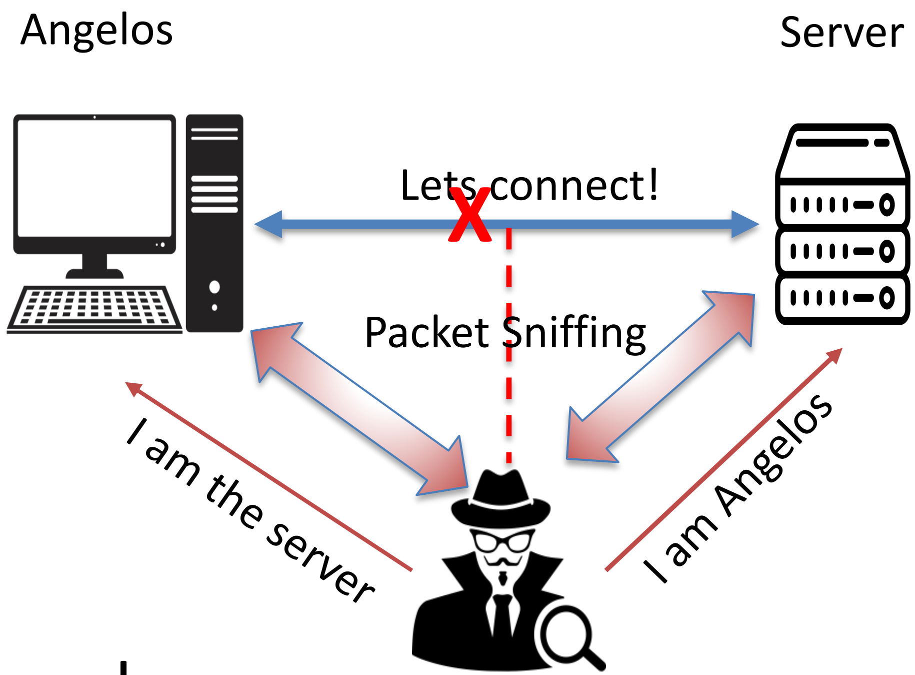
Real Threats…



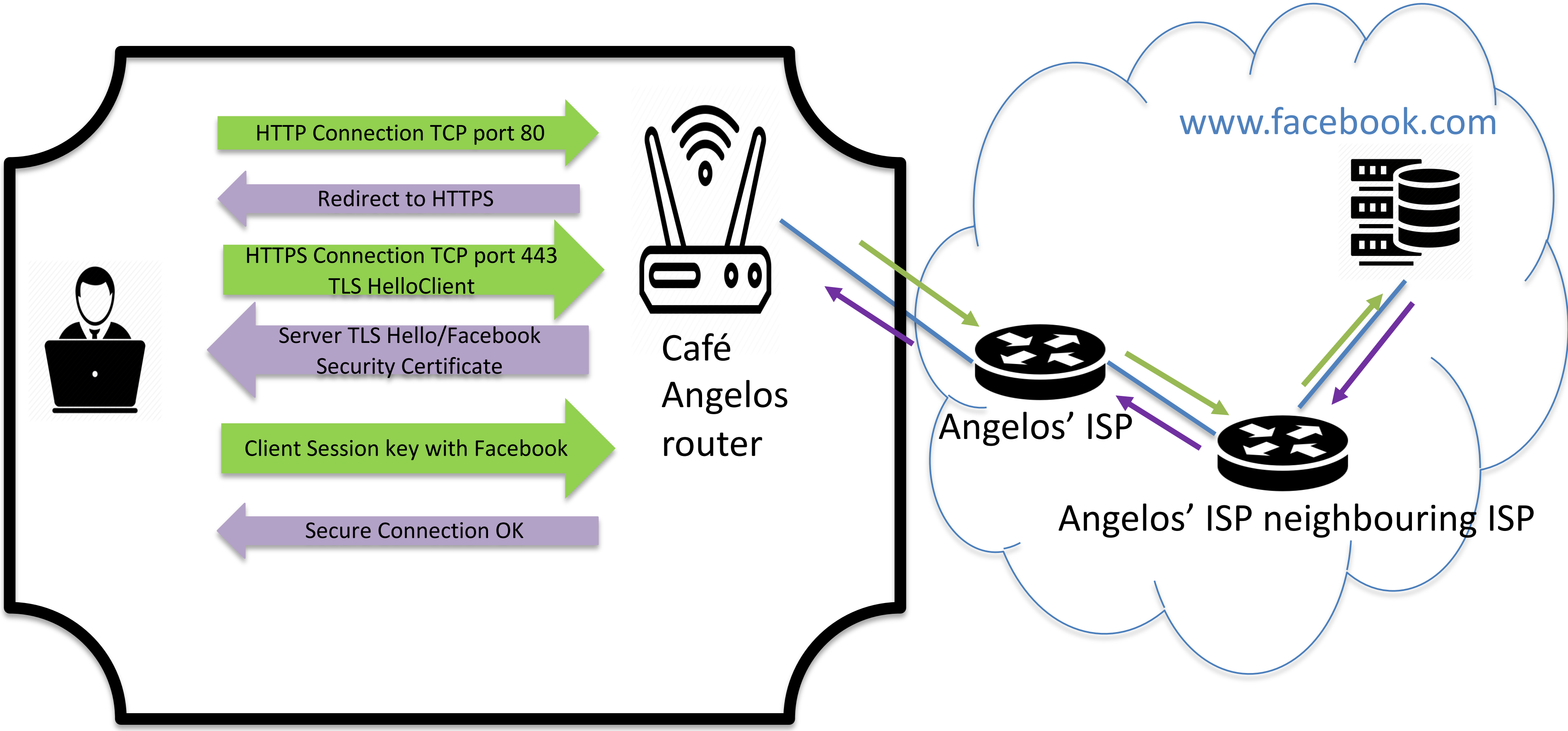
Grossman, J. “The land that Information Security Forgot”, BlackHat Europe 2001

Man-in-the-Middle (*MitM*) Attack

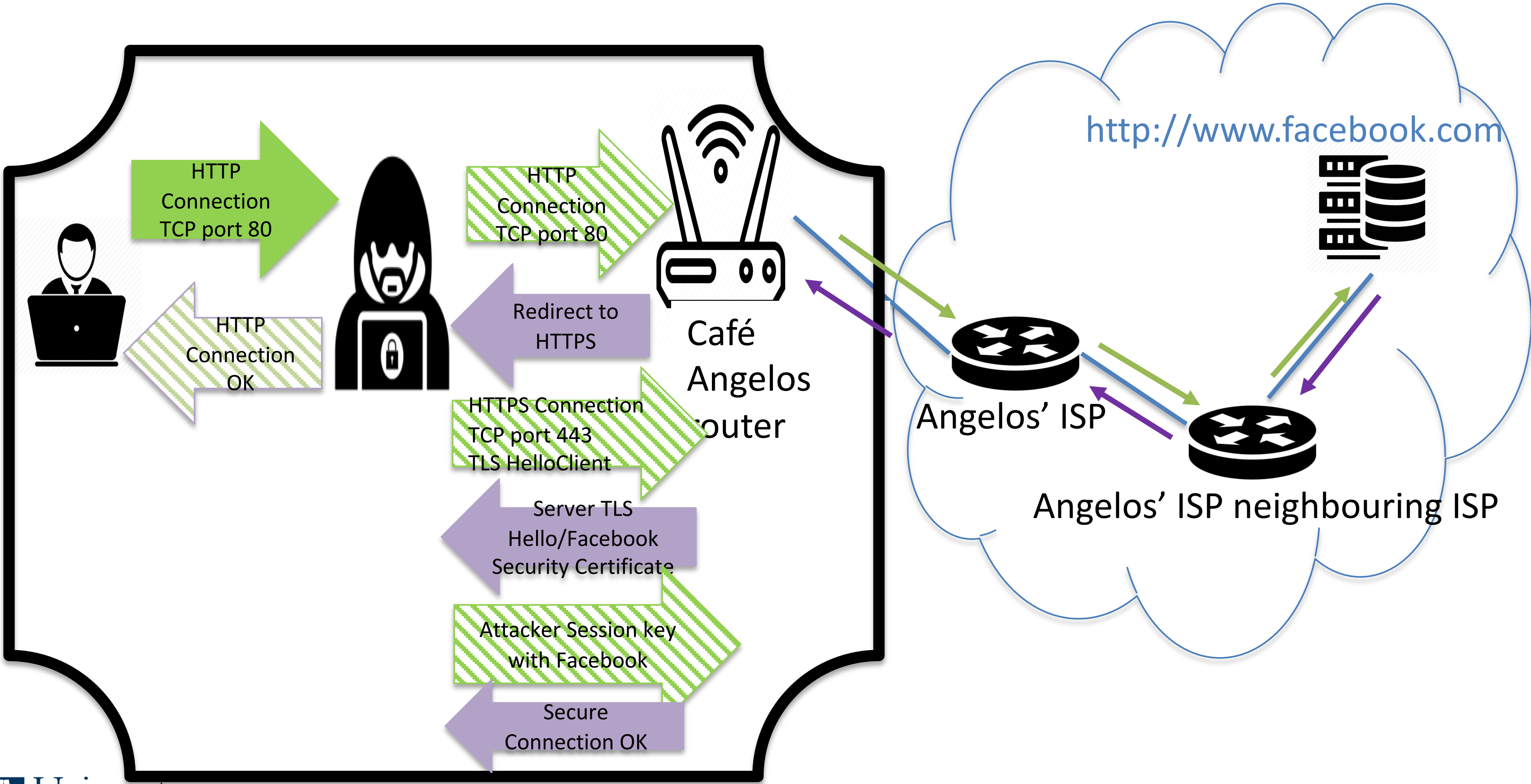
- Many types of MitM:
  - ARP poisoning
  - Port stealing
  - IP address spoofing
  - ICMP redirection
  - SSL hijacking
  - HTTPS Spoofing
  - SSL Stripping
  - ……and the list goes on and on



Example: SSL Stripping


HTTP Connection TCP port 80
Redirect to HTTPS
HTTPS Connection TCP port 443 TLS HelloClient
Server TLS Hello/Facebook Security Certificate
Client Session key with Facebook
Secure Connection OK
Café Angelos router
www.facebook.com
Angelos' ISP
Angelos' ISP neighbouring ISP

Example: SSL Stripping (Cont..)


HTTP Connection TCP port 80
HTTP Connection TCP port 80
HTTP Connection OK
Redirect to HTTPS
Café Angelos router
HTTPS Connection TCP port 443 TLS HelloClient
Server TLS Hello/Facebook Security Certificate
Attacker Session key with Facebook
Secure Connection OK
http://www.facebook.com
Angelos' ISP
Angelos' ISP neighbouring ISP

Aside: But how?

- Packet Sniffers: tools in many OS (especially on GNU/Linux distros or Net/Free/Open BSD)
- SSL stripping/ARP poisoning/SQL injection etc.
- Many offensive hacking tools that can be easily deployed.
- Just have a look at this exhaustive list of tools : https://tools.kali.org/tools-listing

Validating Input Data

- Networked applications fundamentally deal with data supplied by un-trusted third parties
  - Data read from the network may not conform to the protocol specification
  - Due to ignorance and/or bugs
  - Due to malice, and a desire to disrupt services
- Must carefully validate all data before use
- The network is hostile: any networked application is security critical
  - Must carefully specify behaviour with both correct and incorrect inputs
  - Must carefully validate inputs and handle errors
  - Must take additional care if using type- and memory-unsafe languages, such as C and C++, since these have additional failure modes

Example: Buffer Overflow Attack

- Memory-safe programming languages check array bounds
  - Fail cleanly with exception on out-of-bound access
  - Behaviour is clearly defined at all times
- Unsafe languages, such as C and C++, don't check
  - Responsibility of the programmer to ensure bounds are not violated
  - Easy to get wrong – typically results in a "core dump" – or undefined behaviour
- Buffer overflows in network code are one of the main sources of security problems
  - If you write network code in C/C++, be very careful to check array bounds
  - If your code can be crashed by received network traffic, it probably has an exploitable buffer overflow
  - http://insecure.org/stf/smashstack.html

Tips for secure applications

- Do not trust client-side data
  - Check for unescaped special characters
  - Null characters should all be removed
- Make sure that you go through re-authentication procedures (issuing new sessions keys) for your client.

- Some interesting tips can be found here : https://www.w3.org/Security/Faq/wwwsf4.html

Discussion

- Many vulnerabilities already exist over the various OSI layers; this verifies the argument that networks and the Internet were never developed for satisfying security and privacy.
- The Internet : over-engineered with security , does it work?
- Many networked applications written in memory- or type-unsafe languages
  - Many good historical reasons for this, and clearly will take time to replace old deployments with safe alternatives
- Is it justifiable to write new networked code in this way, now that there are safe alternatives?
  - Java, Go, C#, Swift, Rust, …
- As scientists/engineers, we have a duty to use best practices – could you defend your implementation choices?

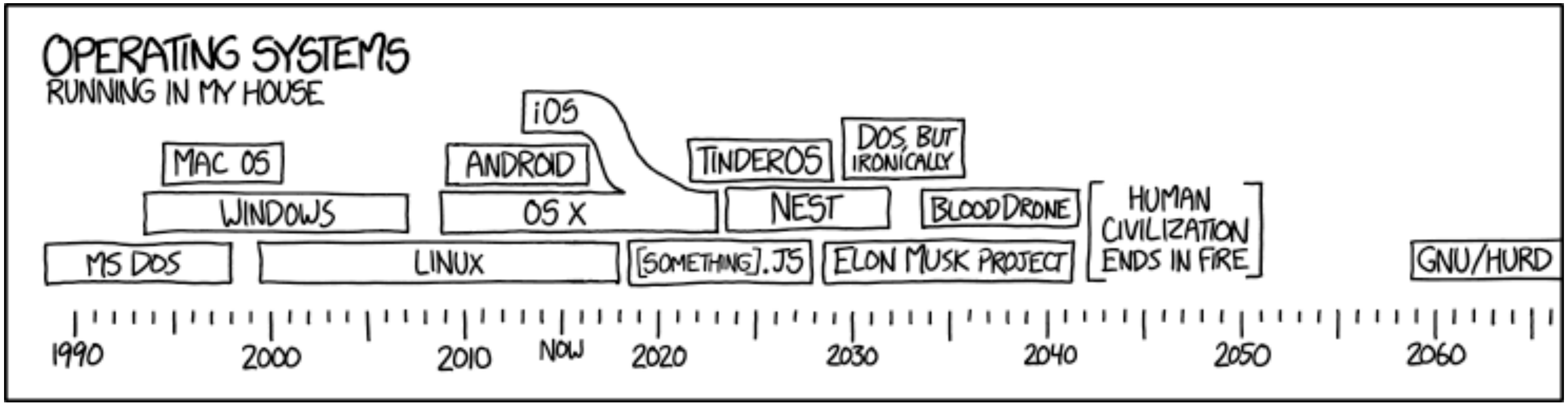
Recommended Reading Material

- You should not read extensively in the suggested chapters below. Material from the first security lecture should suffice (excluding the privacy bit which the GDPR link is provided if you want to know more).

Additional material:

- William Stallings, Lawrie Brown, "Computer Security: Principle and Practice" Chapter 5 & 19 (privacy) (4th Edition), Pearson Education Limited, 2018
- William Stallings, "Cryptography and Network Security Principles and Practices", Fourth Edition, Prentice Hall Pub Date: November 16, 2005
- Online:
  - **GDPR: https://gdpr.eu/data-privacy/**
  - A practical guide (out of many online) for running a MitM: https://linuxhint.com/mimt_attacks_linux/

Coming up next…



Source: https://xkcd.com/1508/