



Security - Lecture 2

Defenses

Clémentine Maurice
L3 ENS - 2020/2021



Today's lecture

- Brief recap about projects
- How to protect data and communications?
 - Authentication and access control
 - Encryption
- Security mechanisms in the operating system

Projects



Projects

Complete list with details: <https://cmaurice.fr/teaching/ENS/>

If you take this class: form groups of 2 and send me your list of the 7 projects by order of preference by **October 5** (clementine.maurice@irisa.fr).

#1: Adversarial machine learning

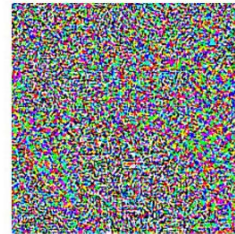
Can you trick a computer into thinking a panda is a gibbon, a turtle, or a plane?

Yes.



“panda”
57.7% confidence

+ .007 ×



“nematode”
8.2% confidence

=



“gibbon”
99.3 % confidence

#2 802.11 fingerprinting

- Wi-Fi devices leave a lot of traces
- These traces can be used to track devices or people
- Observe **network packets in the wild!**



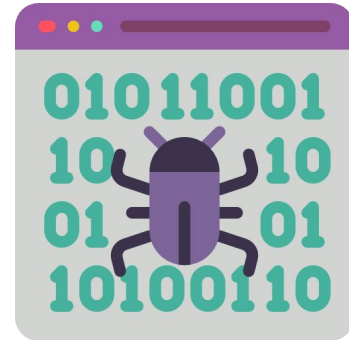
#3 Password cracking

- Passwords are the worst (but they are the best we have)
- Administrators store them unsecurely, users choose weak ones and reuse them
- All of it is a gift for **password crackers (you!)**



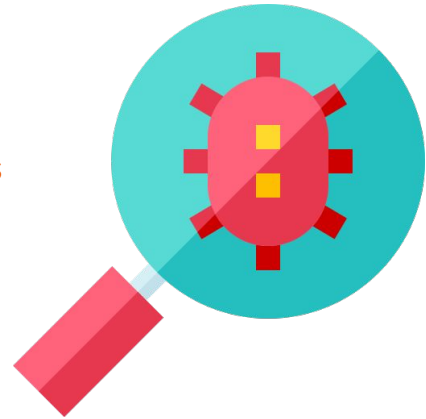
#4 Automated bug finding

- Bugs! They are everywhere and can be dangerous for security
- Discover methods to automatically find them
- **Gotta catch 'em all** (without too much effort)!



#5 Reverse-engineering

- You have a binary, you don't know what it does
- Maybe it's even malicious!
- Learn reverse-engineering techniques through a **series of challenges**



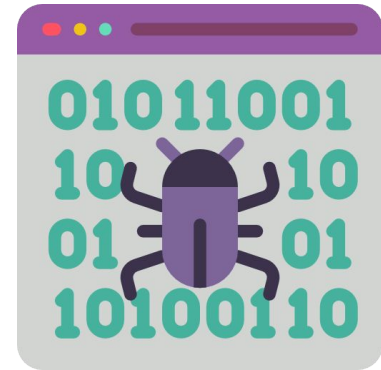
#6 Crypto in the real world

- Crypto is the best! It's maths, so it should be secure
- But even maths can't save you if used badly
- Break all the crypto things (**without any key**)!



#7 Buffer overflows

- Buffer overflows have been around since mid 90s but the vulnerability is still quite common today
- Learn how to **exploit all the things!**



How to protect data and communications?



Authentication and access control



Authentication

- Authentication: **verifying a user's identity**
- Fundamental security building block
 - basis of access control and user accountability
- 2 components: identifier and authenticator
- **Identifier** answers “who are you?”
- **Authenticator** answers “are you really who you say you are?”

Examples of identifiers and authenticators?

Authentication





Authentication

- The “theory”: authentication information can be of 3 different generic types
 - what you **know**: passwords
 - what you **have**: tokens, smartcard, private key
 - who you **are**: fingerprint, retina
- Passwords still the best way to authenticate on a system
- Other mechanisms becoming more popular
 - fingerprint readers
 - two-factor authentication



Authentication: Potential issues

Examples of bad identifiers or authenticators? Potential issues?

Authentication: Potential issues

Examples of bad identifiers or authenticators? Potential issues?

A badge can be shared, lost, or stolen.

More worryingly, you cannot replace your fingerprints...



Hacker fakes German minister's fingerprints using photos of her hands

Jan Krissler used high resolution photos, including one from a government press office, to successfully recreate the fingerprints of Germany's defence minister





Authentication: Storing passwords

- Passwords should **never, ever, be stored in clear**
- **Hash functions**: one-way functions that map data of arbitrary size into fixed-size value
- We hash passwords with **cryptographic** hash functions
 - easy to check the password
 - hard to recover it: hard to revert the hash
 - collision resistant: infeasible to find two different messages with the same hash value
 - small change in the message → new hash value appears uncorrelated with old hash value

there is **hope** $\xrightarrow{\text{SHA1}}$ fae4547e73cc27bbc40694d37a1228042fd89bc4

there is **rope** $\xrightarrow{\text{SHA1}}$ 6cd67725327bb04bb498caecf538d144364e9975



Authentication: Breaking passwords 101

- **Brute force** passwords attacks (a.k.a exhaustive key search)
 - online: try **all possible passwords** to login
 - offline: hash all possible passwords and compare it to stored hash
- Better password cracking
 - **dictionary attacks**: trying all the strings in a pre-arranged listing (= deemed most likely to succeed)
 - rainbow tables
 - tools exist: hashcat, JohnTheRipper
 - password cracking is very easily parallelizable → heavy use of GPUs
- Easy to break passwords: words in **any dictionary**, your user name, your name, names of people you know, substituting some characters (a 0 (zero) for an o, or a 1 for an l), words from book or movies, sequences of keys on a keyboard (QWERTY), all the previous + some numbers (clem223)
- We'll see more about this with Project #3



Authentication: Storing passwords securely

Defending against **brute force**

- slow algorithm, e.g., apply modified DES multiple times
- **salt**, concatenated with the password
 - random but not secret: stored with the hash
 - prevent same passwords to map onto same string
 - makes **dictionary attacks** more difficult: increases the password space

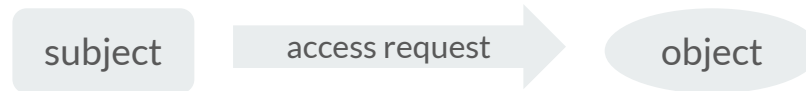


Authentication: Password managers

- Create **random passwords** for each use (website)
- Encrypt the password database with a passphrase
- Can be performed with a file in gnupg or with integrated tools
- Some are cross-platform: phone+laptop synchronized
- Helps to avoid sharing passwords across services → many leaks recently (cf Lecture 1)
- No need to remember them!

Access control

- Access control: the **prevention of unauthorized use of a resource** (incl. prevention of use of a resource **in an unauthorized manner**)
- Elements:
 - **subject**: entity that can access objects → a process representing a user/an application
 - **object**: access controlled resource → e.g., files, directories, hardware
 - **access right**: way in which a subject accesses an object → e.g., read, write, execute, delete, create



Access control matrix

	object 1	object 2	object n
subject 1	allowed ops	allowed ops	allowed ops
subject 2	allowed ops	allowed ops	allowed ops
subject n	allowed ops	allowed ops	allowed ops

capability
list

access control list (ACL)

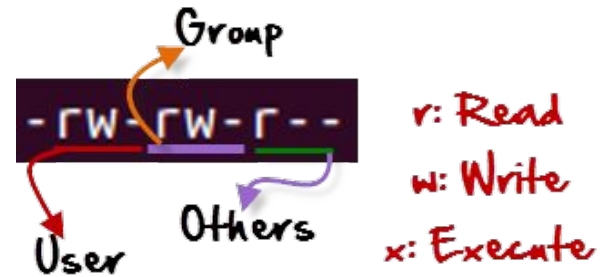


Access control policies

- Discretionary Access Control (DAC)
 - **subjects** can change the policy for **objects they own** → Linux policy
 - problem: admin does not control users who can define any permission on their files + processes inherit of user permissions (can do some damage if malware is run by a privileged user)
- Mandatory Access Control (MAC) → SELinux
 - access policy decided **centrally**, everything that is not allowed is forbidden
- Role Based Access Control (RBAC)
 - access control depending on the **role of a user** (e.g., depending on the position of an employee)
- Rule Based Access Control (RBAC or RB-RBAC)
 - dynamic roles depending on **rules** (e.g., Bob has access to fileA only during certain hours of the day)

Access permissions on Linux

```
% ls -li
total 1474788
11665417 drwxr-xr-x  2 tangerine tangerine Desktop
11796810 drwxr-xr-x 20 tangerine tangerine Documents
11665418 drwxr-xr-x 26 tangerine tangerine Downloads
11665422 drwxr-xr-x 76 tangerine tangerine Music
11665423 drwxr-xr-x 14 tangerine tangerine Pictures
11665424 drwxr-xr-x  7 tangerine tangerine Videos
 4718957 -rw-rw-r--  1 tangerine tangerine archive.zip
 4719576 -rw-rw-r--  1 tangerine tangerine doc1.pdf
 4718948 -rw-rw-r--  1 tangerine tangerine doc2.pdf
11665569 -rw-r--r--  1 tangerine tangerine video1.mp4ds
```



Encryption

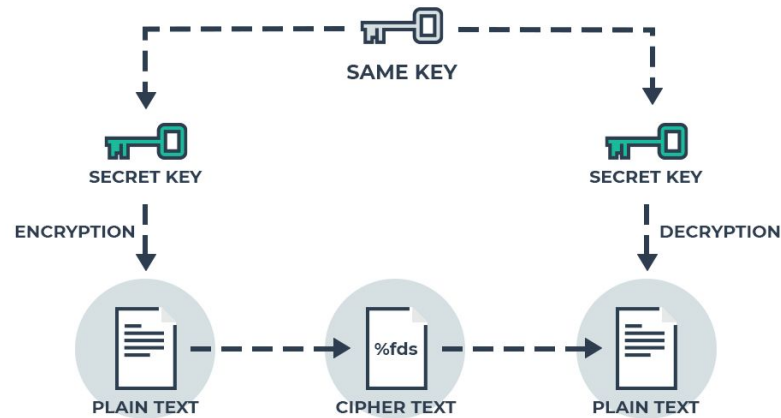


Encryption

- Converting **plaintext** into **ciphertext**
 - converting readable data into something that looks like random
 - requires an **encryption key**
- Won't give details on how the primitives or protocols work
- Modern cryptography: symmetric encryption and asymmetric encryption

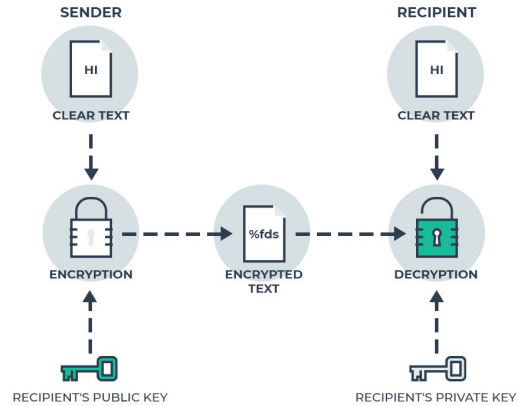
Encryption

Symmetric encryption = same key to encrypt and decrypt



Encryption

Asymmetric encryption = different key to encrypt and decrypt (public key/private key)





Encryption: where should it be performed?

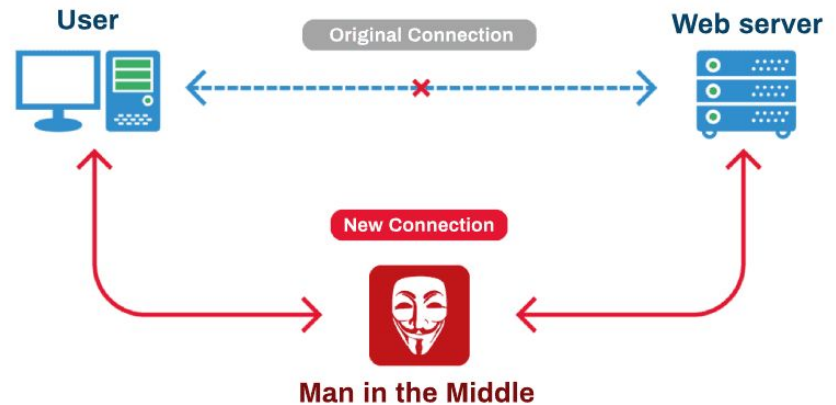
Data can be encrypted "**at rest**," when it is stored, or "**in transit**," while it is being transmitted somewhere else.

Why using encryption in transit?


Avoiding **man-in-the-middle attacks!**

“Popular” attack on Wi-Fi hotspots for insecure connections

If the data in transit is encrypted, the attacker can't use it, or can't send their own messages



Encryption in transit: the case of TLS

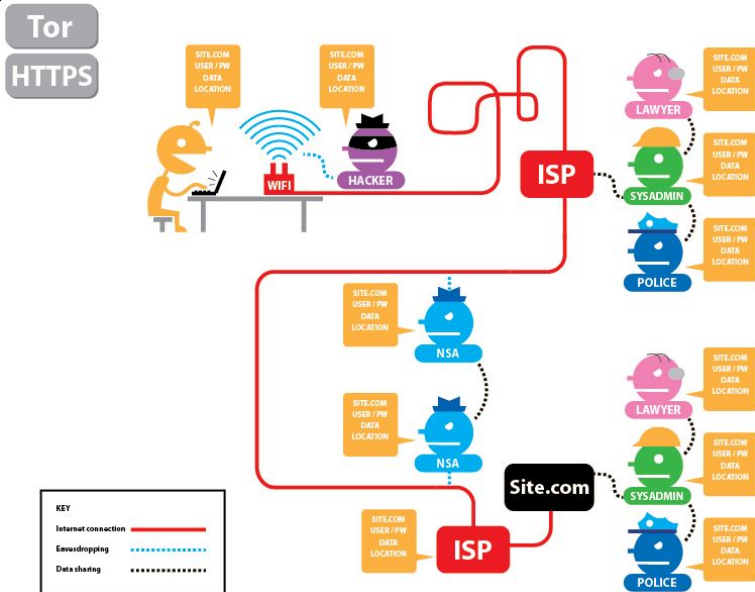
 **HTTPS** = HTTP + TLS

Hypertext Transfer Protocol =
protocol to **transmit messages**
between visitor's browser and
website's server

Transport Layer Security = protocol to
secure communication over a network

What information is protected? From whom?
What information is unprotected?

What does HTTPS do? (1/2)

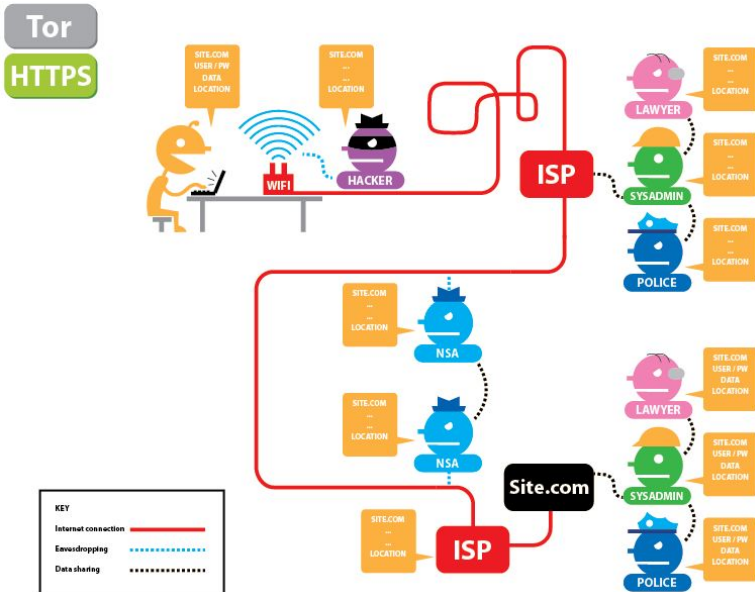


HTTP

- Hacker on local network and ISP: website visited + data/user/password
- Website: your location (IP address)

Source: <https://www.eff.org/pages/tor-and-https>

What does HTTPS do? (2/2)



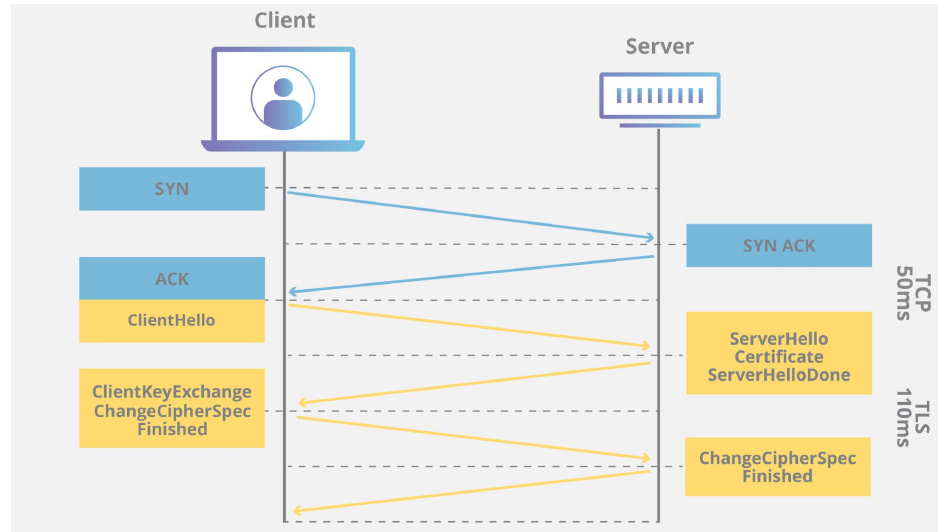
HTTPS

- Hacker on local network and ISP: website visited, but not data/user/password
- Website: your location (IP address) + data

Source: <https://www.eff.org/pages/tor-and-https>

How does TLS work?

- Decide version of TLS
- Decide cipher suites
- **Authenticate the identity of the server** via the server's public key and the Certificate Authority's (CA) digital signature
- Generate session keys

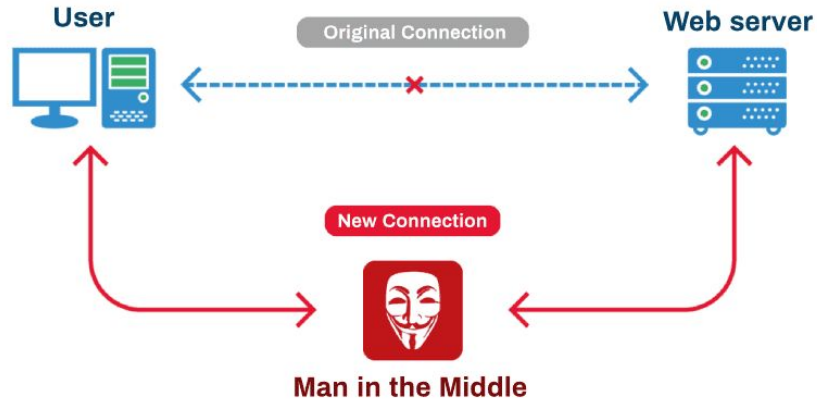




How might TLS fail?

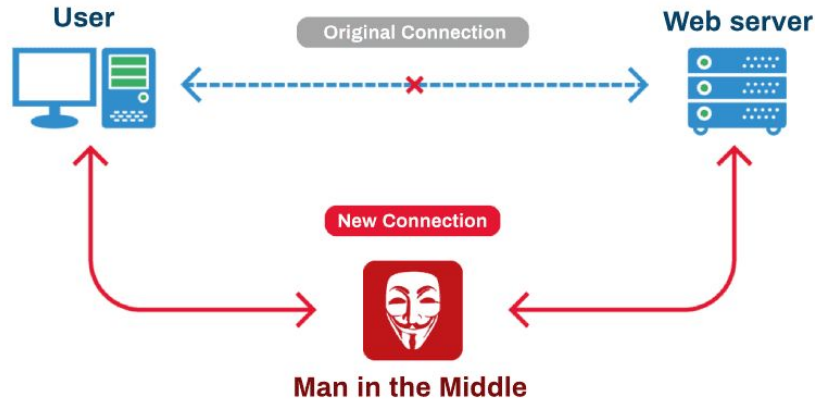
How might TLS fail?

Man-in-the-middle attack on the TLS connection



How might TLS fail?

Man-in-the-middle attack on the TLS connection

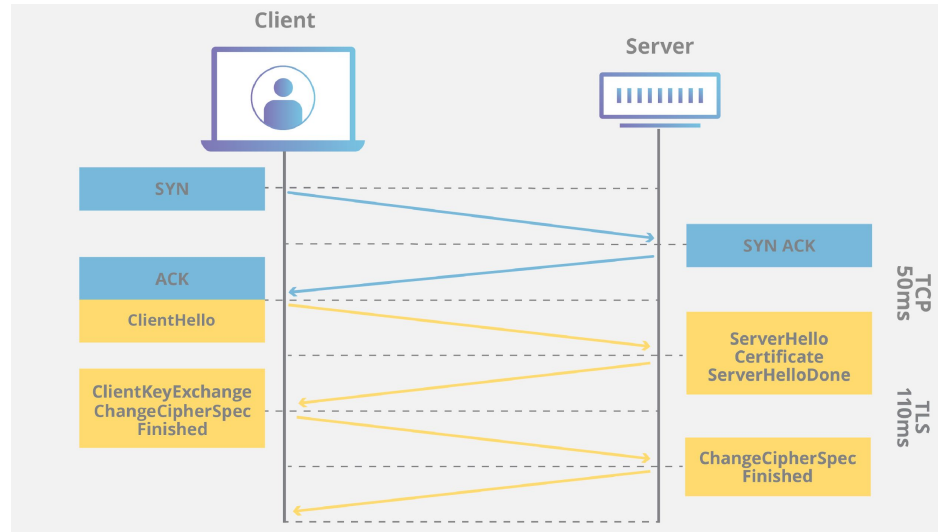


But it's encrypted???

How to intercept messages and impersonate the client or the server?

How does TLS work? (Reminder ;))

- Decide version of TLS
- Decide cipher suites
- **Authenticate the identity of the server** via the server's public key and the Certificate Authority's (CA) digital signature
- Generate session keys



<https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/>

How might TLS fail?

Man-in-the-middle attack on the TLS connection itself

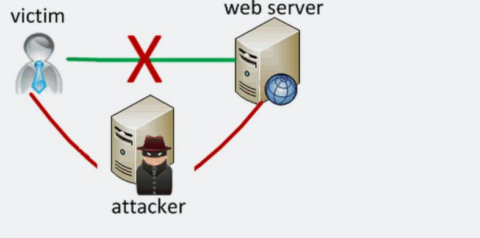
- state actor plant their own root certificate on a trusted Certificate Authority
- operator of a Certificate Authority is corrupted or malicious
- ... or a company thinks it's a smart thing to do (what could go wrong?)

→ The attacker can impersonate the server without the client realizing it

BIZ & IT
Lenovo PCs ship with man-in-the-middle adware that breaks HTTPS connections [Updated]

Superfish may make it trivial for attackers to spoof any HTTPS website.

DAN GOODIN · 2/19/2015, 5:36 PM



333

Lenovo is selling computers that come preinstalled with adware that hijacks encrypted Web sessions and may make users vulnerable to HTTPS man-in-the-middle attacks that are trivial for attackers to carry out, security researchers said.

Your Headphones Might Break The Security of Your Computer



Prof Bill Buchanan OBE [Follow](#)
Nov 29, 2018 · 6 min read ★

Sennheiser has now been pinpointed as have a major security vulnerability in its HeadSetup app. It involves a **self-signed** TLS signature and which Sennheiser placed in the Trusted Root CA Certificate store (or in the macOS Trust Store). This means that this certificate can be used to validate other certificates, as the private key on the certificate could be easily extracted.

Once the private key is derived, it is then possible to sign for maliciously installed applications, as we have a trusted root certificate. The password on the certificate was **SennheiserCC**, and was found by reverse engineering the HeadsSetup application and finding the configuration file :

```
secorvo@ubuntu: /mnt/hgfs/File Lock/HeadSetup
secorvo@ubuntu: /mnt/hgfs/File Lock/HeadSetup$ grep -A5 -B5 -l pass WBCCServer.p
opertles
openssl.server.caConfig = SennComCCCert.pem
openssl.server.verifyMode = relaxed
openssl.server.verifyDepth = 9
openssl.server.loadDefaultCAFile = true
openssl.server.cipherList = ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH
openssl.server.privateKeyPassphraseHandler.name = KeyFileHandler
openssl.server.privateKeyPassphraseHandler.options.password = SennheiserCC
openssl.server.invalidCertificateHandler.name = AcceptCertificateHandler
openssl.server.extendedVerification = false
openssl.server.cacheSessions = true
openssl.server.sessionIdContext = ${application.name}
openssl.server.sessionCacheSize = 100
secorvo@ubuntu: /mnt/hgfs/File Lock/HeadSetup$
```

To prove vulnerability, Secorvo spoofed a Google certificate, along with other audio companies [CVE-2018-17612]:



Encryption in transit

Problems with encryption in transit?



Encryption in transit

Problems with encryption in transit?

- Data is not encrypted when at rest, in the servers between source and destination
- **Do you trust these servers?**

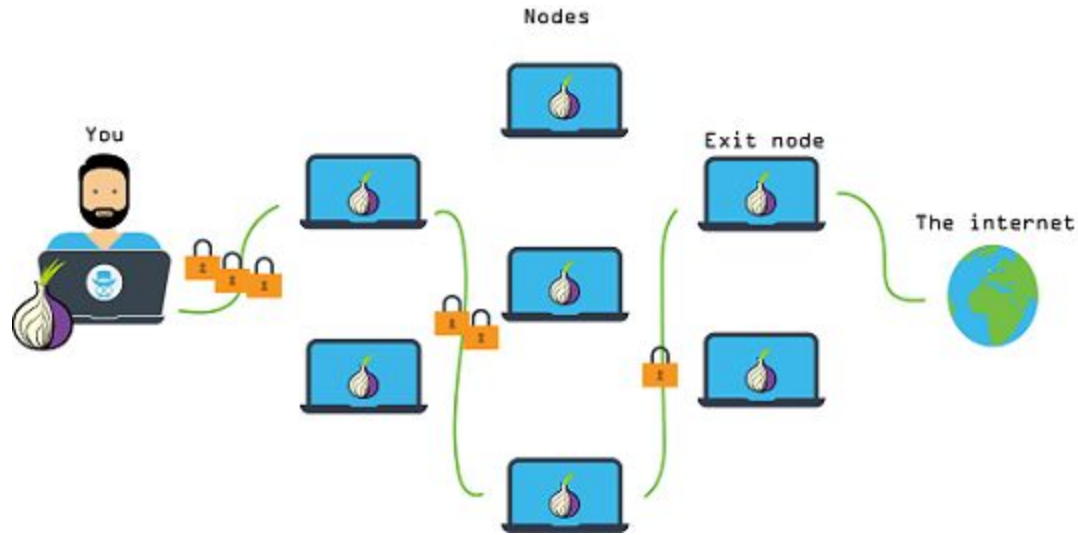


End-to-end encryption

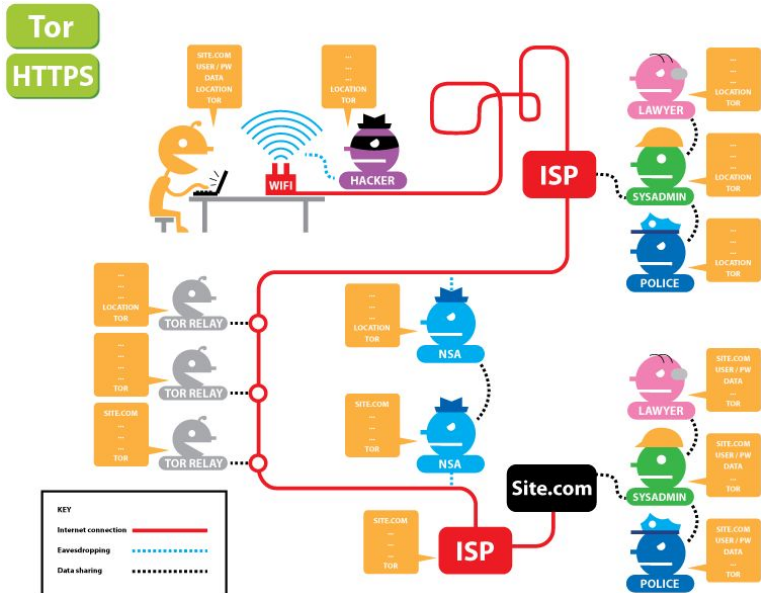
Only the communicating end parties, e.g., the users, **can decrypt** and therefore read the messages

- End-to-end encryption starting to be popular for messaging services, e.g., WhatsApp, Signal...
- Still not the case everywhere
- Important questions:
 - where are the keys stored, and how are they protected?
 - what code runs on the server?
 - what data is actually sent and how is it stored in the server?

Encryption in transit: the case of Tor (1/3)



Encryption in transit: the case of Tor (2/3)

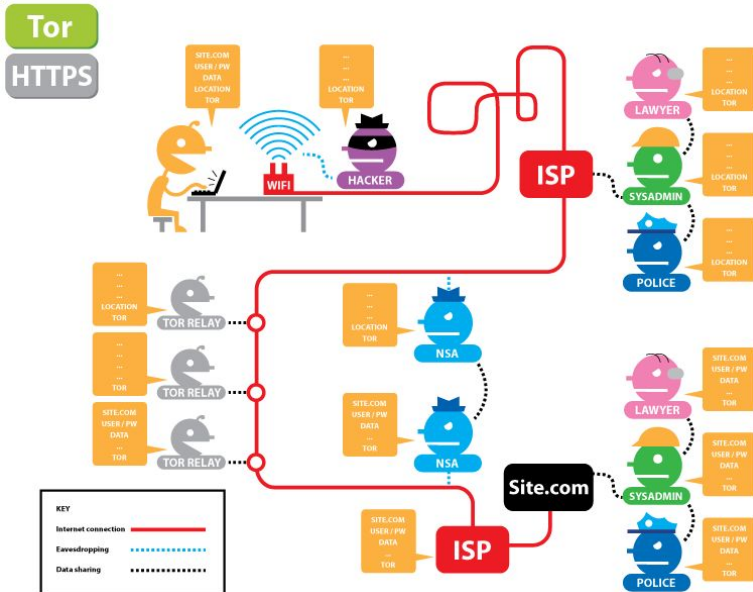


Tor + HTTPS

- Hacker on local network and ISP: not website visited, no data/user/password, but knows you are using Tor
- Website: no location (IP address), but knows you are using Tor

Source: <https://www.eff.org/pages/tor-and-https>

Encryption in transit: the case of Tor (3/3)



Tor, no HTTPS

- Tor exit node: data/user/password
- So does the website ISP and anybody able to intercept ISP traffic (NSA?)
- Not a good idea: not all Tor relays are operated by nice people

Source: <https://www.eff.org/pages/tor-and-https>

System security

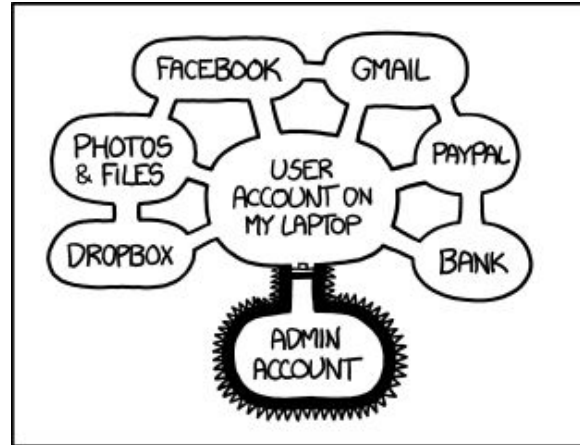




Role of the OS

- Crucial in terms of computer security
- Operating system functionality
 - process management
 - (virtual) memory management
 - file system management
 - I/O management
- Ensures **isolation** between processes and between users, and **access control**
 - keep buggy/malicious apps from crashing/tampering with each other, and from crashing/tampering the OS
 - who can access which resources
- Isolation relies on hardware mechanisms
 - memory management unit
 - protection rings

Even though...



IF SOMEONE STEALS MY LAPTOP WHILE I'M LOGGED IN, THEY CAN READ MY EMAIL, TAKE MY MONEY, AND IMPERSONATE ME TO MY FRIENDS,
BUT AT LEAST THEY CAN'T INSTALL DRIVERS WITHOUT MY PERMISSION.



Virtual memory and MMU

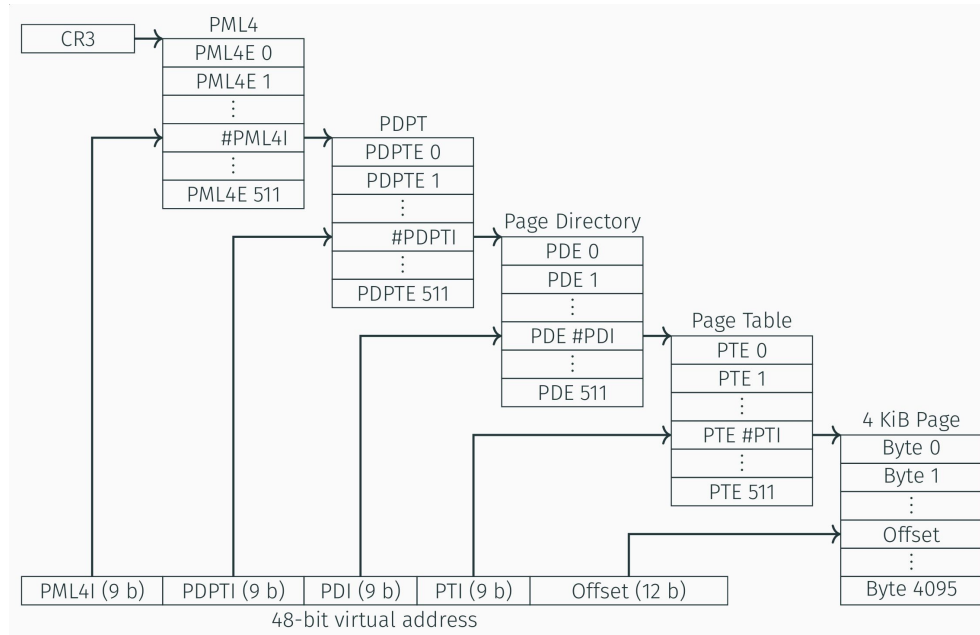
- Introduced in mainframes in 1960s, on PC since Intel 80368 (1985)
- **Isolates** memory spaces of different applications
- Allows applications to use more memory than physically present
- Applications see a **contiguous memory** chunk that may not be contiguous in physical memory
- All **transparent** for applications, managed by the OS
- Applications manipulate **virtual addresses**, DRAM manipulates **physical addresses**
- Memory Management Unit (MMU): hardware unit that translate virtual addresses to physical addresses



Virtual memory and MMU

- Translation is not done with the granularity of one address
 - 2^{32} addresses on a 32-bit system = the translation table would be 16GB
- Memory divided in **pages** (minimum 4KB in x86)
- MMU has a special **cache** called Translation Lookaside Buffer (**TLB**) for recently used translations

Virtual memory and MMU





Virtual memory and MMU

- Page table entry also stores
 - P bit: is the page present?
 - R/W: is the page writable or read-only?
 - U/S bit: is the page accessible in user mode or kernel mode only?
 - NX bit: does the page contain executable code?
- If a process accesses a virtual address for which the page is not valid (P bit not set) or if protections are not correct, **MMU raises an exception** (page fault), OS can, e.g., kill the process
- A process therefore cannot access a physical address for which there is no translation → ensures that **a process cannot access memory of another process, or kernel memory**

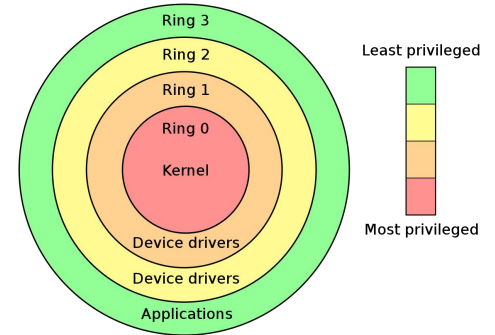


Protection rings in a nutshell

- Introduced in hardware developed for Multics operating system (1973)
- Different software parts will run in **different rings**, with **different privileges**
 - e.g., OS, drivers, libraries, applications
- Most privileged ring (usually called *ring 0*) has access to all functionality and data
- Each ring has access to all data and functionality of the less privileged rings
- Inside a ring, access to functionality and data of a more privileged ring goes through a specific **interface** (e.g., system call)
- How it is implemented and works depends on the CPU and the system that runs on it

Protection rings on x86

- x86 powers most desktop, laptop computers, and servers
- Protection rings in x86 architecture since 80386 processor (1985)
- 4 protection rings in protected mode: *ring 0* (most privileged) to *ring 3* (less privileged)
- In practice, Linux and Windows run with 2 modes: *ring 0* for the **kernel**, *ring 3* for **user applications**
 - OSs rely on the protection brought by paging: for each page table entry, 1 “privileged” bit = 2 “modes” (priv/unpriv)
 - + compatibility with other processors that do not support 4 rings
- Current ring is stored in a special register called CPL (Current Privileged Level)





Protection rings on x86

- Sometimes user code needs to run kernel code (e.g. to interact with hardware)
 - need to change CPL value!
- **Transition** between kernel and user mode
 - software interrupt (store call number in %eax and execute `int 0x80`)
 - SYSENTER/SYSEXIT instructions: faster than interrupts
 - call gates: older and slower, barely used



Protection rings on x86

Began a lot more complicated in recent years due to the apparition of:

- Hardware-based **virtualization**
 - considered as ring -1, but in fact VirtualBox puts the guest kernel OS in ring 1
- System Management Mode (SMM)
 - considered as ring -2
 - power management, system hardware control, managing memory or chipset errors
 - intended to be used by firmware/BIOS



Protection rings on ARMv7-A

- ARM powers most mobile devices (smartphones and tablets)
- 3 protection rings:
 - PL0 (least privileged): User mode, typically used by applications
 - PL1: typically used by the OS
 - PL2 (most privileged): Hypervisor mode
- CPSR (CurrentProcessor Status Register)

Next lecture:

Attacking all the things! Software and hardware attacks

