

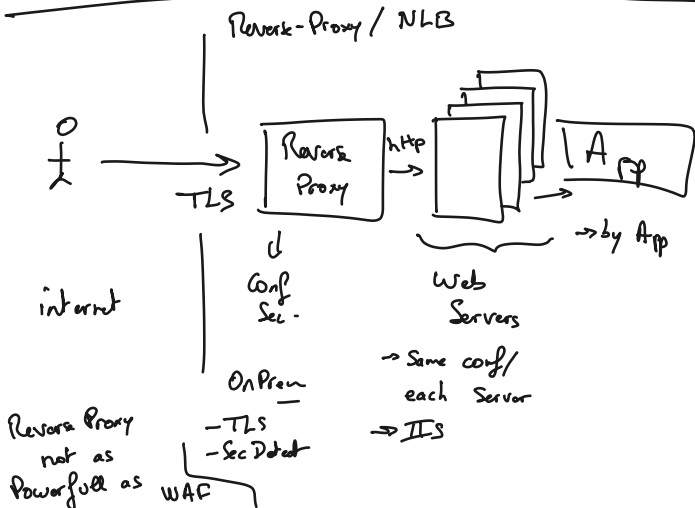
Hello Every One

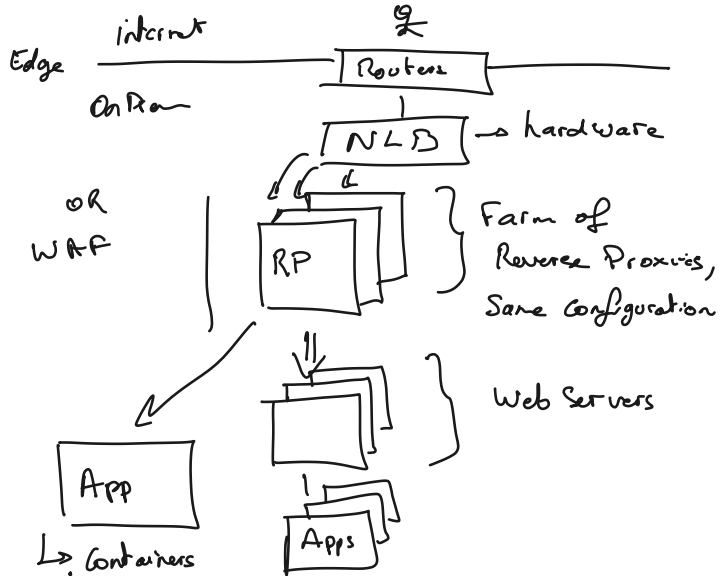
SELinux

.Net CheatSheet : [DotNet Security - OWASP Cheat Sheet Series](#)

All CheatSheets : [Introduction - OWASP Cheat Sheet Series](#)

Proactive controls : [Index Proactive Controls - OWASP Cheat Sheet Series](#)

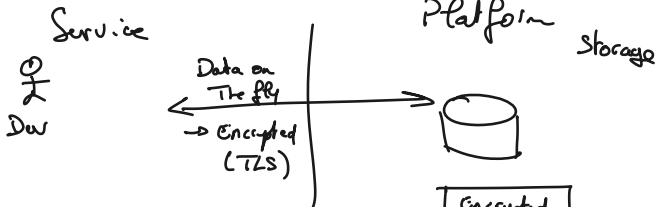




• NLB = just forward Traffic

(- Reverse Proxy = forward by Rules
 - NGINX
 - TRAEFIK
 - WAF = deep inspection for security

Encryption $\rightarrow$ PGP Pub/Private Key



Key $\rightarrow$ encrypt

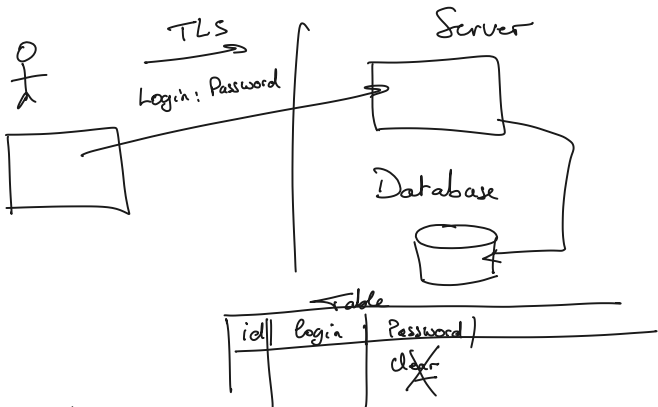
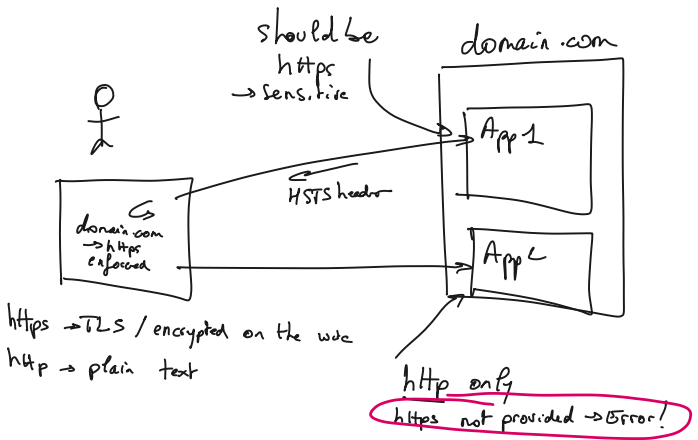
Key $\rightarrow$ decrypt

BitLocker

Encrypted At rest

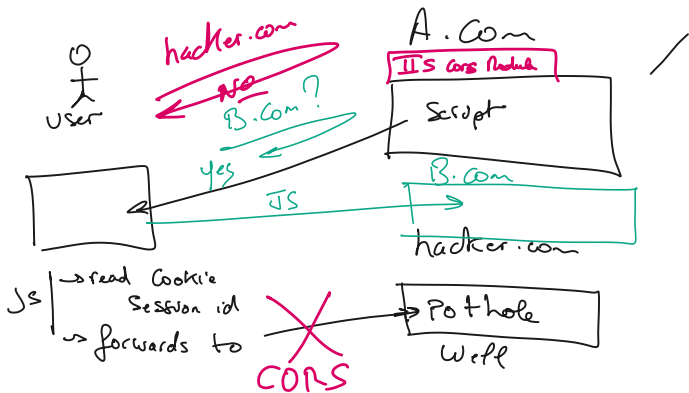
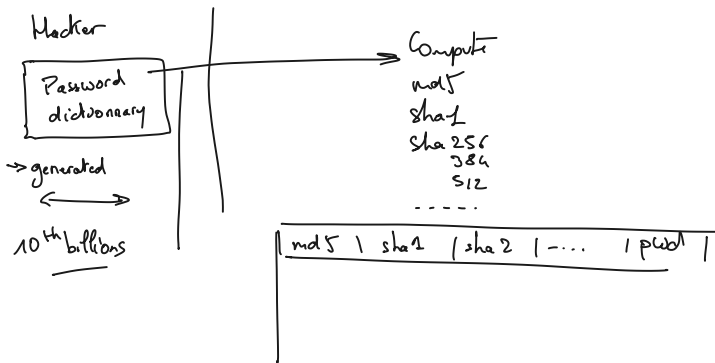
- $\rightarrow$ By the Software
- $\rightarrow$ By the File System
- $\rightarrow$ Block Level

no decryption



→ hashing the password → Trustable? NO

Rainbow Tables



All Security headers : [OWASP Secure Headers Project - OWASP](https://owasp.org/www-project-secure-headers/)

CVE

Traditional..

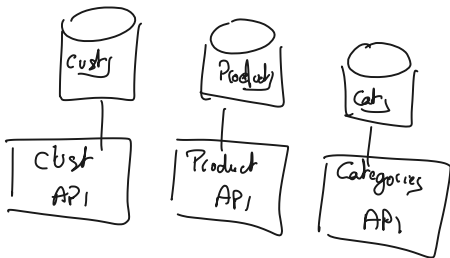


Referenced DB
Table
Relations - joins

→ Consistency oriented

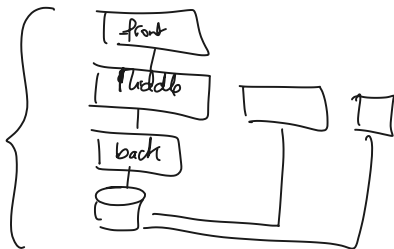
Big Data NoSQL DB
Every Pattern in Referenced
is an Anti-Pattern

→ High Availability oriented

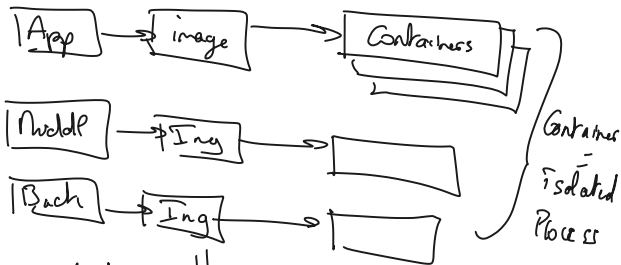


nTier arch.

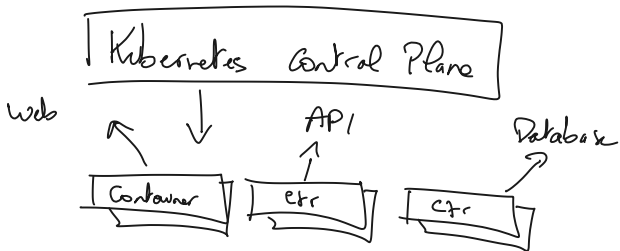
VMs

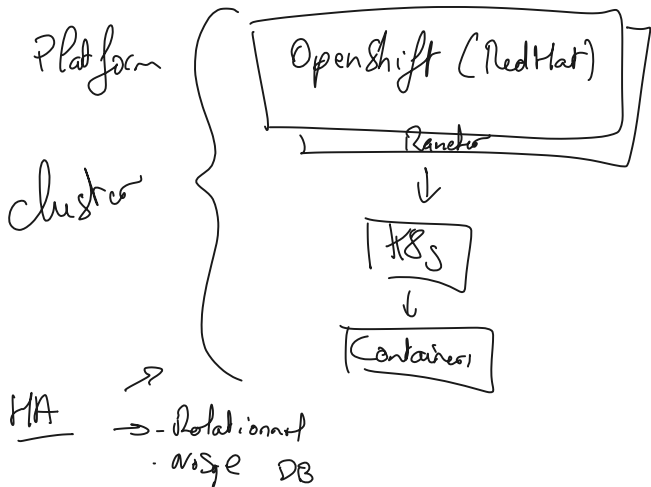


Containers



Production = cluster



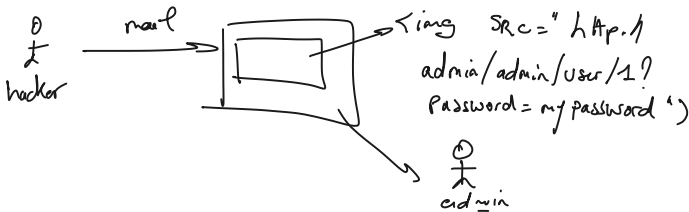
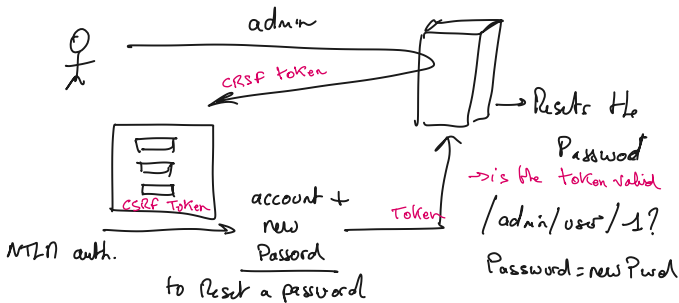


MSrc is a Pattern (architecture)
Implementation / Framework — Istio
— RH as Mesh

Istio → open source on K8s MSF FW.

- Encryption — Fly — Storage
- Auth enforcement
- Log Centralisation + Monitoring
- ACL

CSRF



Linux:

ACLs

basic
level

{ U ser
g roup
o thers

Read
write
execute → d → browsing the dir.

FACLs

other

ACLs + more specific to user
groups

user (process)

↓ access file/folder/device

Cancel

!OK

SELinux → OK → Proceed
to next step

base

(

ACL

→ OK → Proceed

↓ !OK

extension

(

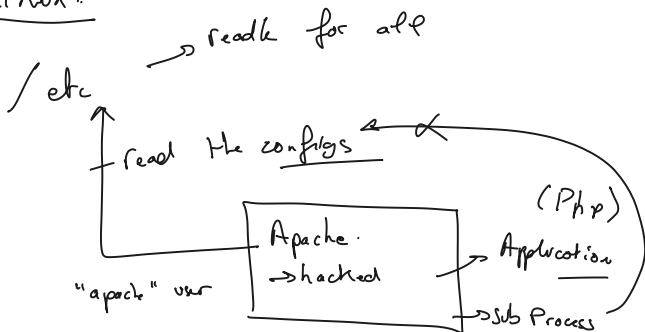
FACLs

→ OK → Proceed

↓ !OK

? ...

Linux.



Confinement

→ Context

→ which user /
which process?

"SELinux
confinement"

apache user & apache process =

read to /var/http/html <only>

hack tries to write in /var/http/html

→ forbidden

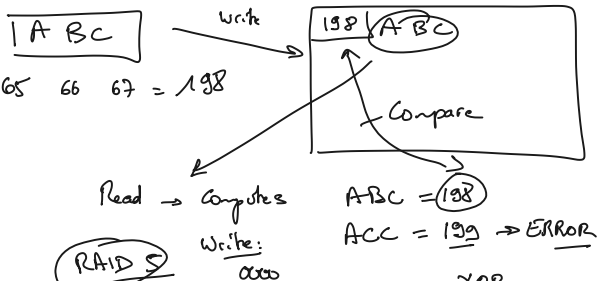
write in /var/data → forbid

there are no strategies

HASH

hash functions md5
sha

check file integrity = CRC = check sum



RAID

RAID 5

$$0 \oplus 0 = 0$$

$$1 \oplus 0 = 1$$

$$1 \oplus 1 = 0$$

(+)

XOR
AND
OR

$\rightarrow$ incr - Speed
- Reliability

Write 000 1111

~~0000 1111~~

0000 1111

(+)

0000 0000

0000 1111

Repair

Collisions

$$f(x) = y \rightarrow \text{out}$$

in

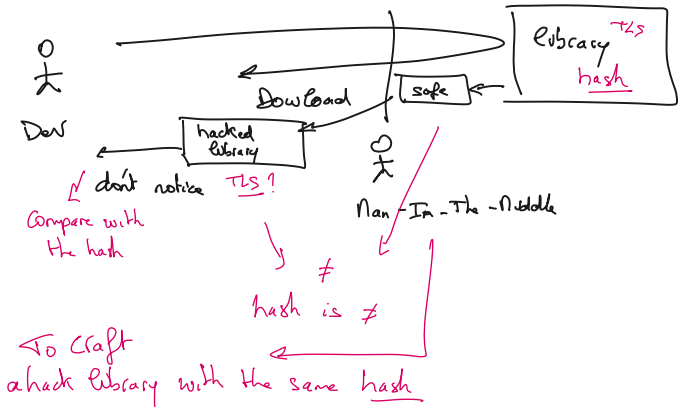
$$f(x_1) = f(x_2)$$

Collision

$$ABC = 198$$

$$ACB = 198$$

CRC



✓ Password → text printable string

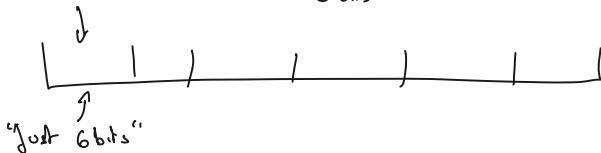
✓ Key → very long printable machine generated Password

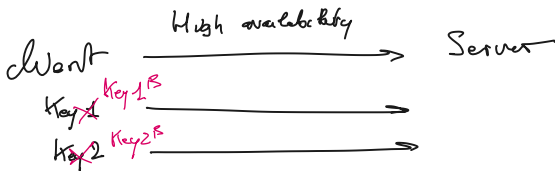
Certificate → Authentication x509 format.

✓ Secret = either a password or a key
86 bits → 1 byte

bytes

Printable characters = 1 64 characters out of 256
86 bits ↓ 6 bits ← Possibilities 86 bits





| ~~Key 1~~ $\rightarrow$ Compromised $\rightarrow$ Key 1^B
~~Key 2~~ $\rightarrow$ Key 2^B

Non printable "passwords"

authentication certificates -

exported as a file
 $\downarrow$
 binary

as a text file
 $\downarrow$
 base64 encoding

- o csr : format pkcs10
- o pem, key, cert, crt : base64 dont multipart, pub et priv keys
- § -----BEGIN CERTIFICATE-----
- § -----BEGIN CERTIFICATE REQUEST-----
- § -----BEGIN PRIVATE KEY-----
- § -----BEGIN ENCRYPTED PRIVATE KEY-----
- § key : format pem, uniquement la clef privée
- § cert, cer, crt : Windows
- o pkcs12, p12, pfx : PKCS#12 (binaire)
- o der : flux binaire du certificat.
- o p7b, keystore : inclut le chemin de certification.
- crl : liste de révocation.

Password + Salt

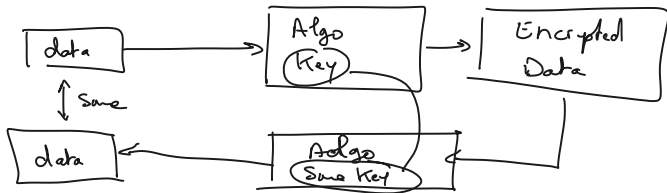
User Table

Salt → Random text string
Server side generated

id	Logon	hash	salt
		f5310....	yrtz2d

Symmetric Encryption

1 Key / password / ...



1990's

DES

→ brute force crack within months

DES → 1 password → obsolete

→ 3DES → 3 passwords

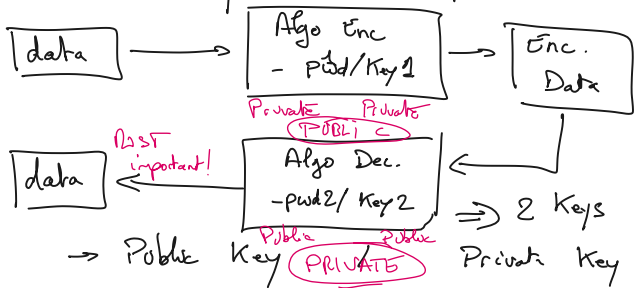
1 password | 1st Part | 2nd | 3rd |

3DES → 1st Part → 2nd → 3rd → Encrypted Data
DES DES DES

2nd Symmetric algo = AES

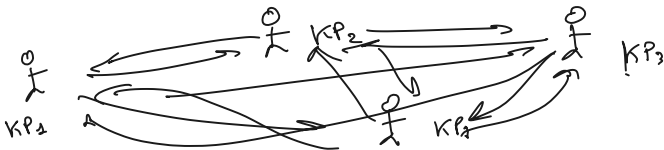
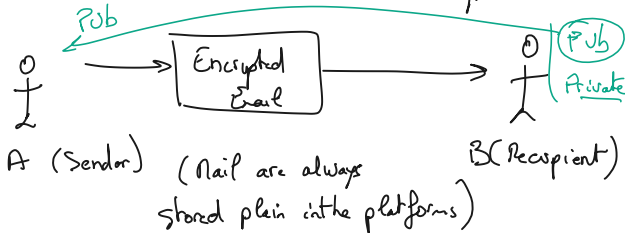
Commonly used with
Certificates -

Asymmetric Encryption



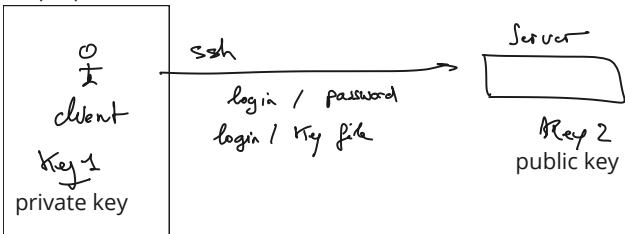
Ex:

SNTP / $\rightarrow$ Signature
 $\rightarrow$ Encryption



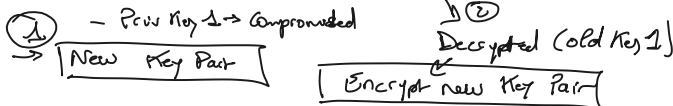
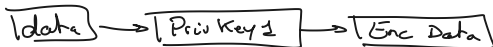
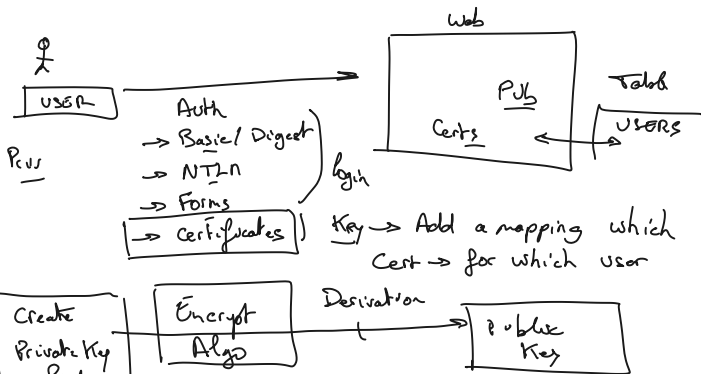
laptop

SSH



→ bitlocker / GPG on windows
→ ext luks

Web Site Auth



Master Key

Algo

Key Pair +

Master Key (Decrypt)

→ "I can open all the doors"

Ex:

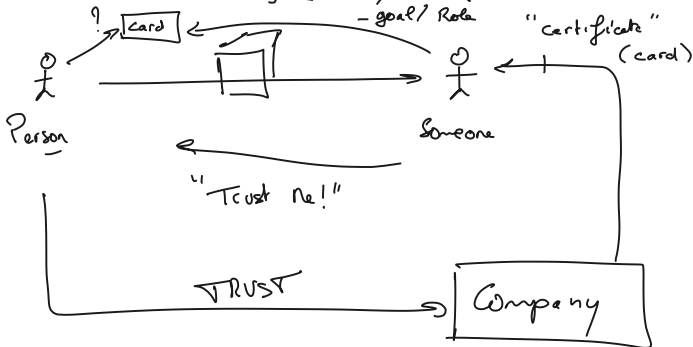
EFS

Certificates

The Point is Trust

→ validity: Time, chain of Trust

- goal / Role



Public Key Infrastructure

Process

① Build the PKI → Concretely = certificates

heart bleed

openssl command line

Windows AD Certification Services

↳ web site, API, ...

other tools ...

② Build Final Certificates, use

③ To Trust them, Deploy The Root/
Intermediate Certificate Authority
to the client trust

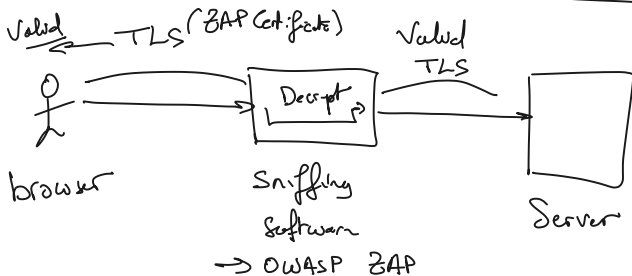
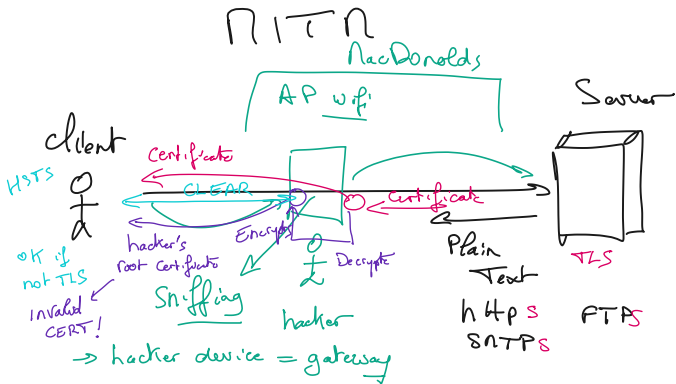
Edge } mmc/certs
• net

java → jks file
(java KeyStore)

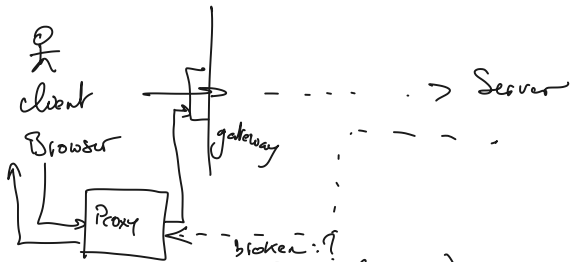
/etc/ — → global to

Postman → own certificates
chrome ↗
firefox ---- ↗

/var/lib/java/version/ — .jks



internet



Proxy $\Rightarrow$ - ~~Caching~~ (go's)
- Security $\rightarrow$ Contents?

TLS

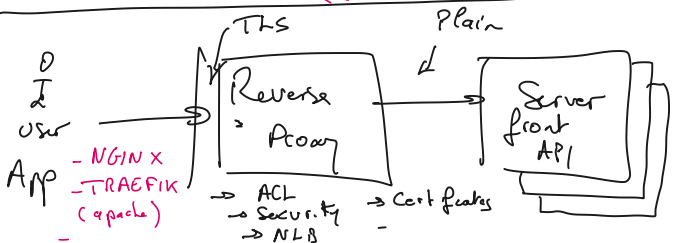
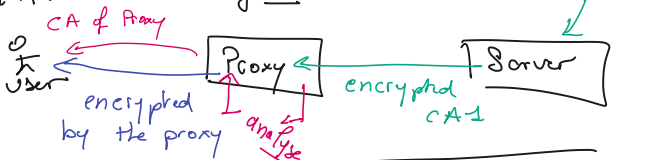
$\rightarrow$ Dest / Source

$\hookrightarrow$ IP address

$\hookrightarrow$ black listing
white listing

OR: Setup

PKI and Certificates $\Rightarrow$ "MITN"

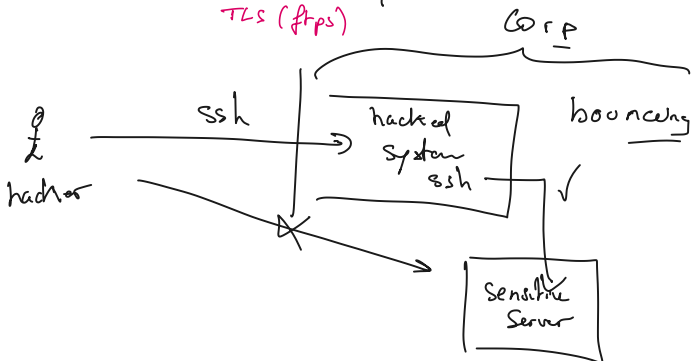
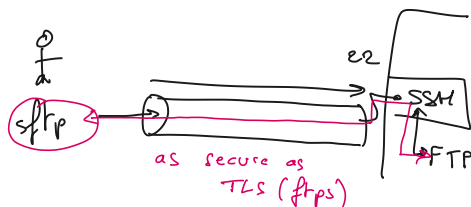


$\text{http} \xrightarrow{+\text{TLS}} \text{https}$
 $\text{ftp} \longrightarrow \text{ftps}$
 $\text{smtp} \longrightarrow \text{smtps}$

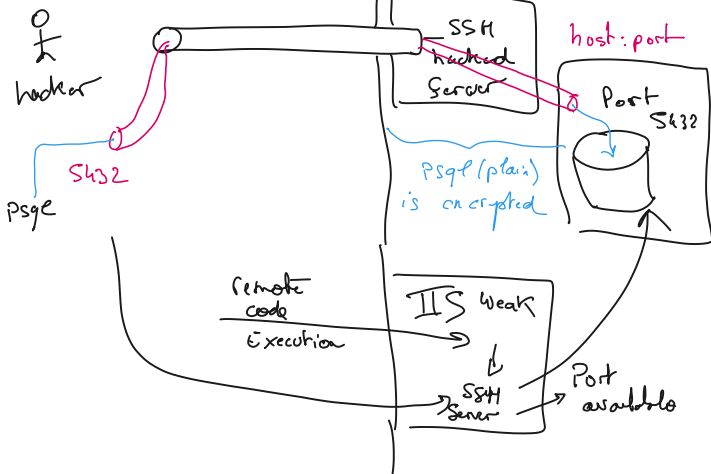
SSH as powerful as TLS

$\text{sftp} \longrightarrow \text{ftp on ssh}$

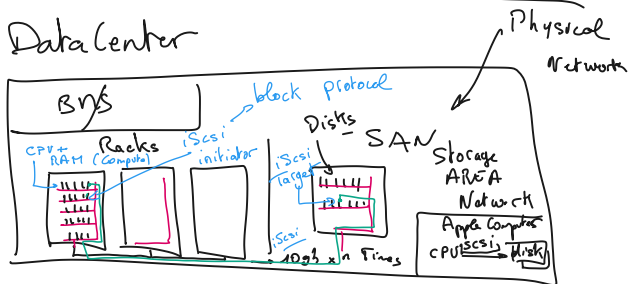
Secure shell



SSH Tunnels



Data Center



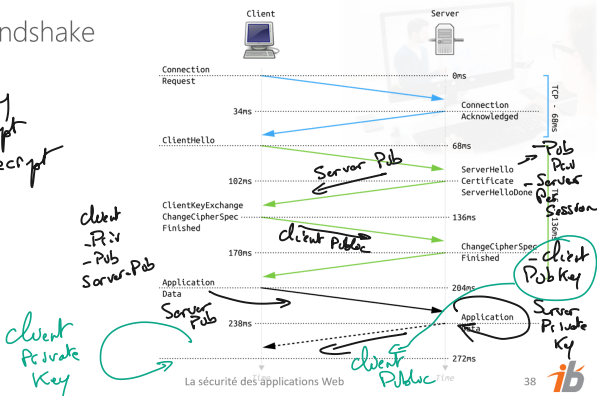
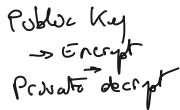
NAS: Network Access Storage

↳ Filesystem level:

NFS

SMB (Windows Shares)

TLS Handshake



Synth

TLS

no TLS:

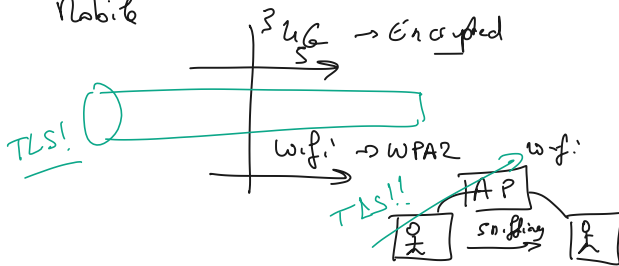
- No encryption
- No Cert
- Sessions (K:1
Memory ← Ses

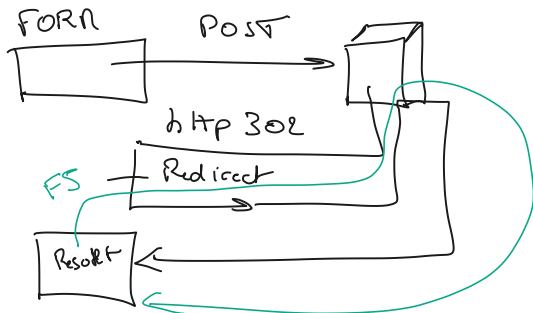
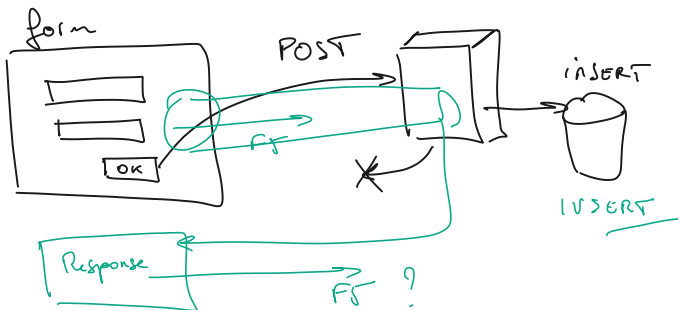
- + Server Certificates PKI
(Domain: EV, DV, -)

- Session Server Key Pair + Client Pub Key / Session Key: Values

+ CPU | encrypt } - hardware
| Decrypt } encryption -

Nabit





```

echo "Installing Docker:"
sudo apt update
sudo apt -y upgrade
sudo apt install -y docker.io openjdk-14-jdk git curl
sudo systemctl enable docker
sudo systemctl start docker

echo "Installing docker-compose:"
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

echo "*****"
echo "Preparing dwaa:"
sudo docker network create zapnet
sudo docker container run -d --name dwaa --net zapnet -p 1080:80 -p 3306:3306 -e MYSQL_PASS="Pa55w.rd" citizenstig/dwaa
sudo docker container run -d --name zap --net zapnet -u zap -p 1081:8080 -p 1090:8090 owasp/zap2docker-stable zap-webswing.sh
echo "*****"

echo "*****"
echo "Preparing mutillidae II:"
sudo mkdir -p /opt
cd /opt
sudo git clone https://github.com/webpwnized/mutillidae-docker
cd mutillidae-docker
sudo docker-compose up -d
echo "Connectez-vous sur http://localhost pour Mutillidae"
echo "*****"

echo "*****"
echo ""
echo "..."
echo "mysql password : Pa55w.rd ( port 3306 )"
echo "..."
echo "SCAN with ZAP : "
echo "# docker run --name zap --rm --net zapnet -i owasp/zap2docker-stable zap-cli quick-scan --self-contained --start-options '-config api.disablekey=true'"
echo "http://dwaa.zapnet:1080"
echo "# Afficher les paramètres de zap-cli"
echo "sudo docker run --rm --net zapnet -i owasp/zap2docker-stable zap-cli"

echo "BASELINE : "
echo "# docker container run --name zap --rm --net zapnet -t owasp/zap2docker-stable zap-baseline.py -t http://dwaa.zapnet:1080"
echo "sudo docker container run --rm --net zapnet -t owasp/zap2docker-stable zap-baseline.py"
echo ""
echo "..."
echo "Owasp zap démarré."
echo "..."
echo "Connect to DVWA : http://localhost:1080, setup the database, and connect as admin / password, and go setting up the security level to LOW"
echo "ZAP WebGui : http://localhost:1081/zap"
echo "proxy address : http://localhost:1090/"
echo "# sudo docker container run -d --name zap --net zapnet -u zap -p 1081:8080 -p 1090:8090 owasp/zap2docker-stable zap-webswing.sh"
echo "scan app from zap : http://dwaa.zapnet"
echo ""
echo "Télécharger et installer owasp : https://www.zaproxy.org/download/"
echo "small Rockyou : https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/Leaked-Databases/rockyou-20.txt"
echo "sudo docker container docker run --name juice-shop -p 3000:3000 bkimminich/juice-shop"
echo ""
echo "Owasp Juice Shop disponible sur http://localhost:3000"

```