

Group 6

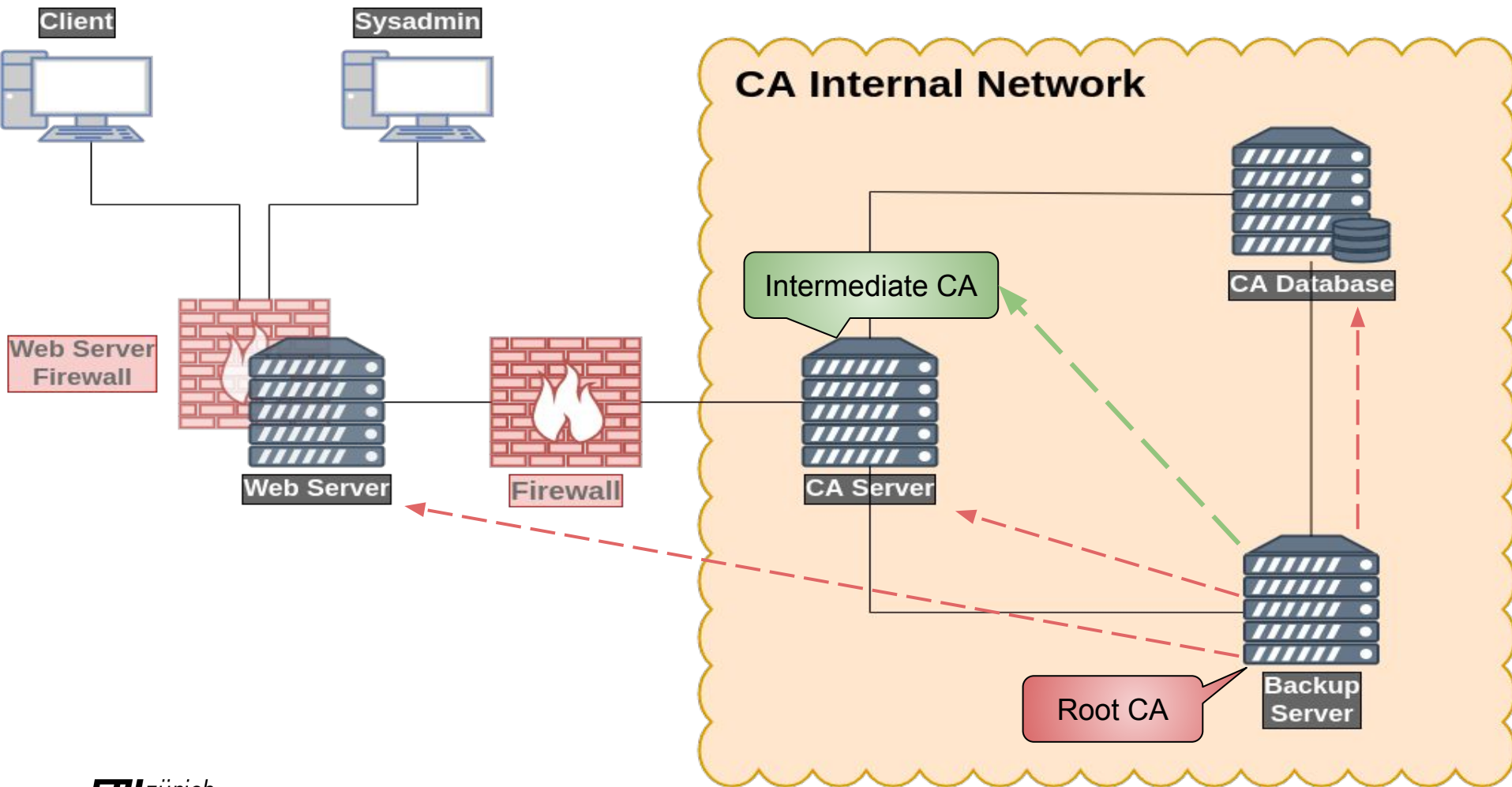
System presentation

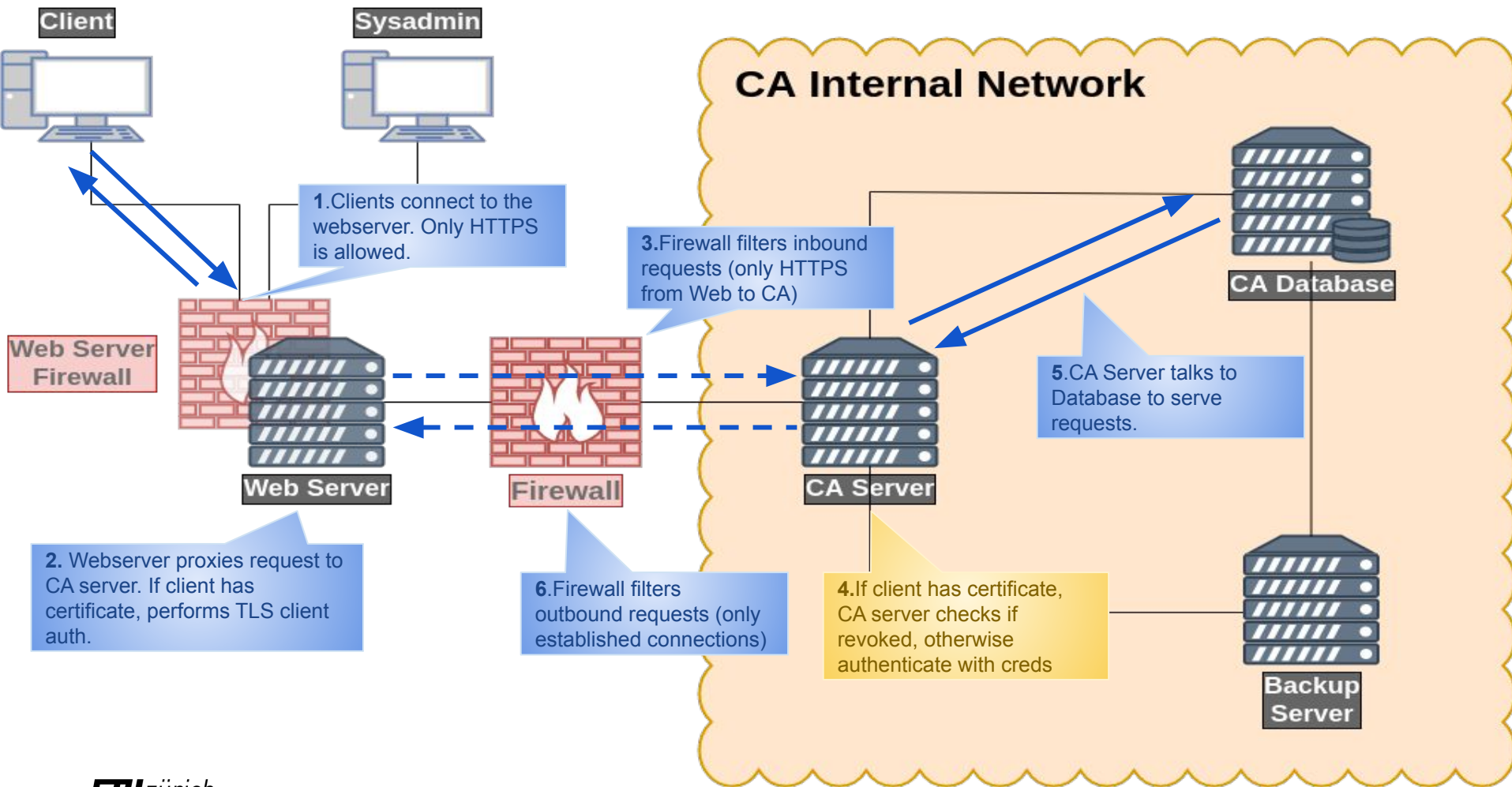
Lucie Hoffmann,
Francesco Intoci,
Elvric Trombert,
Lucas Rollet



Agenda

1. Architecture Highlights
2. Backup
3. Security Design
4. Backdoors
5. Final remarks





Backup

- rsyslog
 - *from all machines (webserver, caserver, firewall and database)*
 - *Done over TLS in real time*
 - *Include nginx access and error logs from webserver and caserver*
- database backup every day
 - *using sftp user dbackup*
 - *chrooted*
 - *no ssh login*
- private key backup straight after their generation
 - *encrypted on the caserver then sent to the backup*
 - *using sftp user cabackup*
 - *chrooted*
 - *no ssh login*
- Database and client private key backups are copied to a directory inaccessible to the sftp users. Overwriting of the copied files is prevented

Review of Security Mechanisms - Architecture Level

- **Network:**

- **Compartmentalization:** network segmentation
- **Separation of Privileges:** each machine has a specific role
- **Least Privilege:** machines traffic is restricted for their specified role
- **Zero-Trust Model:** no implicit trust between machines

- **Data protection:**

- **In transit:** encryption everywhere (TLS, SSH)
- **At rest:** encryption of clients private keys and system configurations

Review of Security Mechanisms - Application Level

- **Protection against common web vulnerabilities:**
 - **XSS:**
 - React automatically escapes XSS payloads
 - Session cookie with *HTTP Only* flag
 - **CSRF:**
 - Session cookie with *Same-Site strict* attribute
 - allow only POST requests for state-changing requests
 - **SQLi:** prepared statements
 - **Auth via JWT:** not tamperable (HMAC-SHA256)
- **Availability:** Basic rate limiting for heavy tasks (DoS mitigation)

Easy - JWT “none” attack

Algorithm

HS256

JWT String

Verified!

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1aWQ0IjphMyIsImZlcnRtaW4iOiJGYWxzZSI6ImV4cCI6IjIwMjEtMTktMTIgMDc6Mzc6NTkifQ.83r9YZzhU-3Kjbd-i99k_01GHnhmociZP_zBfLLcx7M

Header

{
 "typ": "JWT",
 "alg": "HS256"
}

Payload

{
 "uid": "a3",
 "isAdmin": "False",
 "exp": "2021-19-12 07:37:59"
}

Signing key

i

Base64 encoded

NTNv7j0TuYARvmNMmWXo6fKvM4o6nv/aUi9ryX38ZH+L1bkrdD10bOQ8JAUmHCBq7Iy7otZcyAagBLHVkvYaIpnMuxMARQ97jUVG16Jkpkp1wXOPsrF9zwew6Tp
czyHkHgX5EuLg2MeBuIT/qJACs1J0apru00JCg/g0tkjB4c=

JWT “none” attack - exploit

Algorithm

none

JWT String

Verified!

eyJ0eXAiOiJKV1QiLCJhbGciOiJIub251In0.eyJ1YWQiOiJhMyIsImZlcnRtaW4iOiJUcVliIiwiaXhwIjoimjAyMS0xOS0xMiAwNzozNzo1OSJ9

Stripped signature

Header

Payload

{
 "typ": "JWT",
 "alg": "none"
}

{
 "uid": "a3",
 "isAdmin": "True",
 "exp": "2021-19-12 07:37:59"
}

Signing key

i

Base64 encoded

NTNv7j0TuYARvmNMmWXo6fKvM4o6nv/aUi9ryX38ZH+L1bkrd10bQ8JAUmHCBq7Iy7otZcyAagBLHVkvYaIpmMuxMARQ97jUVG16Jkpkp1wXOPsrF9zwew6Tp

Hard - Kleptography - discovery

Listed in */robots.txt*



To the attention of NSA Agent Michael J. Wiener
The AES_CBC_128 key we used is: odelkyjvtqasmww

Useful hint ;)

```
cacert.pem
server > cert > cacert.pem
1  -----BEGIN CERTIFICATE-----
2  MIIEHTCCAwWgAwIBAgIU00Q09DUG6ozAacb54r1Sn0f1W1zWdQYJKoZIhvcNAQEL
3  BQAwfzELMAkGA1UEBhMC08gxCzAJBgNVBAGMA1ZEMREwDwYDVQQHDAhMYXVzYW5u
4  ZTEQMA4GA1UECgwHSU1vdmllczELMAkGA1UECwwCQ0ExEDA0BgNVBAMMB2ltb3Zp
5  ZXMxHAdBgqhkiG9w0BCQEWEGFkbWluQGLtb3ZpZXMuY2gwHhcnMjExMTI1MTYx
6  NDE2WhcnMjExMTI1MTYxNDE2WjB/MQswCQYDVQQGEwJDS0ELMAkGA1UECwVkb3Qx
7  ETAPBgNVBACMCExhdXNhbm5LMRAwDgYDVQQKADJTW92aWVzMQswCQYDVQQLDAJD
8  QTEQMA4GA1UEAwwHaW1vdmllczEfmB0GCSqGSIb3DQEJARYQYWRTaW5AaW1vdmll
9  cy5jaDCCASwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAL0qWgQcUA4P78Rh
10 FJMDvSgKkaSMeVmfiTKlsDD5Ltf2nBQyIEGqvwpiaC0HB3Z80TKEBdpF0137s6
11 l0sYmtU1+VGEonI+QalwZMgI3grNUCvw9EJATov7UDRLuiT90Pm7703s4w1HDCS
12 zzMmW1zBd9SaQL83LYhrxXUHVB28GN3naytg5tRil19FuWzqQo7v6hdKvDSEvIf
13 y6KK82du3Iclunill0S0u+uqX4Y+hRypChfkmLssGqoB0pctvWFtd1Ee/wghnSUp
14 JazpFRwRlXpNjC7QMaIloIZKiZqnT89/Y6IJeCKjP2FI0dua3zVETAXaCtCkV
15 NRIHhBECawEAAAOBkDCBjTAdBgNVHQ4EFgQU3MMR3WiiCadY/37D8TEkYt2Tssw
16 KgYDVR0RBcWmYiIAH1vdmllc4ILd2ViLm1tb3ZpZX0CCSuaW1vdmllczAfbgNV
17 HSMGDAWgBTcwHdaKIplj/fsPxMOSTK3Z0yzAPBgNVHRMBAf8EBTADAQH/MAAG
18 A1UdDwEB/wQEAwIBhjANBgqhkiG9w0BAQsFAA0CAQEAAM3tJPgZHBGNZzdFytZ
19 /V4KcZ5Qw0U1nTR5ry4Gr9L08LSAU/yAP68+E0BvV4wwz74Gj5z0HBIPQ8KIUG6d
20 +f9PNwA9AxVvIKtTxPRLLLU0Y12VQtg++vjsP/qBsXc820nFRaoMjMj+pw75SBL
21 O/C18Rtk/7Z2Mhw/ANqt4ximgJSY59054j5nCl85yXKGiUjTsvPGNRupj07iKUN
22 u134yVJNuv94+Z8EykdAhASKmfPKTMgnAFGC/BRUDD7I0Kop6xi+9fqPL5aC4Qv
23 Rwb3MFwa/0Apl7tI+qAW5ivBPwgu29V2LWvLc1pS9LWhdBivRrSTzikfDuzrs'
24 /g==
25 -----END CERTIFICATE-----
26 Key: odelkyjvtqasmww
27 Secret:
28 24390971199819214072010642145947439461630915859168075314231965995645118096637852273798455949964301793169166343
29 81250832319009257960386970346678492231136937675583989942677885977266331862971777130447445740567381251676666165
30 60570087351969543904738278691853289987883725035482343921835490224741673815985609208239068763958948209797582648
31 01931960668465890132217757182060682771940241436914224351206034940635337021277071349359622354651263652356277524
32 34032468714115668010432926283580103568623062772189239459071965497451413081830197801182582662614161620799387319
33 868259732501277016596978255850956738368316696539485900632873949182
```

This is not shown if you use OpenSSL!

Kleptography - exploit

- **SETUP:**
 1. Generate RSA parameters as usual (N, e, d)
 2. Generate a small $|\delta| < |N|/4$ which is also invertible $\text{mod } \phi(N)$
 3. Compute $\varepsilon = \text{inv}(\delta, \phi(N))$
 4. Encrypt ε using AES_CBC_128 with NSA key (this is “secret” in the certificate...)
- **ATTACK:**
 1. Get ε by decrypting the secret with the leaked key.
 2. Apply Wiener’s low exponent attack to get δ (*it’s an algorithm that allows you to retrieve the secret key if it is “small” enough*)
 3. Now you have a multiple of $\phi(N)$: $\varepsilon \cdot \delta - 1$
 4. You can now factor N by known algorithms (yes, this is similar to Miller-Rabin test):
 - Divide $\phi(N)$ by 2 until odd: get s
 - Raise a random base to s : get b
 - If b is not 1, keep squaring it until you get 1
 - If you get 1, then you know that $\text{pow}(b, 2) - 1 = 0 \text{ mod } N$
 - Simply take $\text{gcd}(b+1, N)$ to find (hopefully) a non trivial factor of N

Room for improvement

- Weird generation of certificate with bash + OpenSSL may have lead to larger attack surface
(probably a versioning problem with python cryptography, certificates were not parsed in Browser)
- Probably too much time spent on frontend (we had fun diving into React though)
- Logging via ssh tunnelling
- SSH login using SSH proxy



~\$: echo 'Thanks for the attention'
~\$: sudo wish --Merry_Xmas!