Laborator de criptografie – Infrastructura de chei publice (PKI)

1 Scopul lucrării

Criptografia cu chei publice este baza comunicațiilor sigure de azi. Din păcate, este supusă atacurilor de tipul om-la-mijloc atunci când o parte trimite cheia sa publică unei alte părți. Problema fundamentală este că nu există o cale ușoară de verificare a proprietarului unei chei publice, adică, fiind dată o cheie publică și presupusul proprietar, cum ne asigurăm că respectiva cheie aparține proprietarului presupus? Infrastructura de chei publice (engl. Public-Key Infrastructure (PKI)) este o soluție practică pentru această problemă.

Obiectivul de învățare din această lucrare este ca studenții să obțină experiență nemijlocită în lucrul cu infrastructura de chei publice. După efectuarea lucrării, studenții vor înțelege mai bine cum funcționează PKI, cum este folosită pentru a proteja Web și cum pot fi înfrânte atacurile de tip om-la mijloc. În plus studenții vor putea înțelege rădăcina încrederii din PKI și ce probleme pot apărea dacă încrederea în rădăcină este compromisă Lucrarea acoperă următoarele subiecte:

- Criptarea cu cheie publică, infrastructura de chei publice (engl. Public-Key Infrastructure (PKI))
- Autoritatea de certificat (engl. Certificate Authority (CA)), certificate X.509 și Ca rădăcină
- Apache, HTTP și HTTPS
- Atacurile de tipul om-la-mijloc

2 Desfășurarea lucrării

2.1 Setarea mediului

2.1.1 Configurarea containerului și comenzi.

Vă rugăm să descărcați fișierul `Labsetup.zip` pe VM-ul dvs. de pe Moodle, să-l dezarhivați, să intrați în directorul `Labsetup` și să utilizați fișierul `docker-compose.yml` pentru a configura mediul de laborator. Explicații detaliate ale conținutul acestui fișier și toate fișierele `Dockerfile` implicate pot fi găsite din legătura la **manualul de utilizare - SEED Manual for Containers**. Dacă este prima dată când configurați un mediu de laborator SEED folosind containere, este foarte important să citiți manualul de utilizare.

În cele ce urmează, enumerăm câteva dintre comenzile utilizate frecvent legate de `Docker` și `Compose`. Cum vom folosi aceste comenzi foarte frecvent, am creat aliasuri pentru ele în fișierul `.bashrc` (în SEEDUBuntu 20.04 VM furnizat de noi).

```
$ docker-compose build # Build the container image
$ docker-compose up # Start the container
$ docker-compose down # Shut down the container
// Aliases for the Compose commands above
$ dcbuild # Alias for: docker-compose build
```

```
$ dcup # Alias for: docker-compose up
$ dcdown # Alias for: docker-compose down
```

Toate containerele vor rula în fundal. Pentru a rula comenzi pe un container, avem adesea nevoie să obținem un shell pe respectivul container. Mai întâi trebuie să folosim comanda "`docker ps`" pentru a afla ID-ul containerului și apoi să folosim "`docker exec`" pentru a lansa un shell pe acel container. Am creat aliasuri pentru ele în fișierul `.bashrc`.

```
$ dockps // Alias for: docker ps --format "{{.ID}} {{.Names}}"
$ docksh <id>// Alias for: docker exec -it <id> /bin/bash

// The following example shows how to get a shell inside hostC
$ dockps
b1004832e275 hostA-10.9.0.5
0af4ea7a3e2e hostB-10.9.0.6
9652715c8e0a hostC-10.9.0.7
$ docksh 96
root@9652715c8e0a:/#
// Note: If a docker command requires a container ID, you do not need to
// type the entire ID string. Typing the first few characters will
// be sufficient, as long as they are unique among all the containers.
```

Dacă întâmpinați probleme la configurarea mediului de laborator, vă rugăm să citiți secțiunea "Common Problems" din manual pentru a afla posibile soluții.

Configurarea DNS. În acest document, folosim `www.bank32.com` ca exemplu pentru a arăta cum să configurați un server web HTTPS cu acest nume. Va trebui să folosiți alt nume pentru lucrarea dvs. Dacă nu este specificat numele de către instructori, studenții ar trebui să includă numele de familie și anul curent în numele serverului. Spre exemplu, dacă John Smith face acest laborator în 2022, numele serverului ar trebui să fie `www.smith2023.com`. Nu trebuie să dețineți acest domeniu; trebuie doar să mapați acest nume la adresa IP a containerului adăugând următoarele intrări în `/etc/hosts` (prima intrare este necesară, în caz contrar, exemplul din această lucrare de laborator nu va funcționa):

```
10.9.0.80 www.bank32.com
10.9.0.80 www.smith2020.com
```

2.2 Sarcini de efectuat

2.2.1 Sarcina 1: Deveniți o autoritate de certificat (CA)

O autoritate de certificat[1] este oe entitate de încredere care emite certificate digitale. Un certificat digital certifică proprietatea asupra unei chei publice de către subiectul numit din certificat. Câteva CA-uri comerciale sunt tratate ca CA-uri rădăcină; ; VeriSign este la momentul scrierii lucrării cea mai mare asemenea autoritate. Utilizatorii care doresc să obțină certificate digitale emise de către CA-urile comerciale trebuie să plătească pentru aceasta.

În această lucrare de laborator avem nevoie să creăm certificate digitale, dar nu vom apela la un CA comercial. Vom deveni noi o CA rădăcină și vom folosi această CA pentru a emite certificate pentru alte entități (d.e. pentru server). Spre deosebire de alte certificate, care sunt de obicei semnate de o altă CA, certificatele CA rădăcină sunt auto-semnate. CA-utile rădăcină sunt de obicei preîncărcate în majoritatea sistemelor de operare, browserelor de web și a altui software care se bazează pe PKI. Certificatele CA-urilor rădăcină sunt de încredere în mod necondiționat.

[1] engl. Certificate Authority (CA)

Fișierul de configurare `openssl.conf`**.** Pentru a folosi `OpenSSL` la crearea de certificate este necesar un fișier de configurare. Fișierul de configurare are de obicei extensia `.cnf`. Acest fișier este folosit de trei comenzi `OpenSSL`: `ca`, `req` și `x509`. Pagina de manual a lui `openssl.conf` se poate găsi folosind Google search. De asemenea, se poate obține o copie a fișierului de configurare de din locația `/usr/lib/ssl/openssl.cnf`. După ce ați copiat acest fișier în directorul curentă trebuie să creați câteva subdirectoare așa cum se specifică în fișierul de configurare (vedeți secțiunea `[CA default]` a fișierului):

Listing 1: Setarea implicită a CA

```
[ CA_default ]
dir             = ./demoCA              # Where everything is kept
certs           = $dir/certs            # Where the issued certs are kept
crl_dir         = $dir/crl              # Where the issued crl are kept
database        = $dir/index.txt        # database index file.
#unique_subject = no                    # Set to 'no' to allow creation of
# several certs with same subject.
new_certs_dir   = $dir/newcerts         # default place for new certs.
serial          = $dir/serial           # The current serial number
```

Pentru fișierul `index.txt`, creați pur si simplu un fișier gol. Pentru fișierul `serial`, puneți un singur număr reprezentat ca șir de caractere în fișier (d.e. 1000). O dată ce ați setat fișierul de configurare `openssl.cnf`, puteți crea și emite certificate.

Autoritatea de certificat (CA). Cum am descris mai înainte avem nevoie să generăm un certificat auto-semnat pentru autoritatea noastră de certificat. Aceasta înseamnă că avem încredere totală în această autoritate și că certificatul ei va servi ca certificat rădăcină. Puteți folosi următoarea comandă pentru a genera certificatul auto-semnat pentru CA-ul nostru:

```
openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 \
-keyout ca.key -out ca.crt
```

Vi se vor cere mai multe informații și o parolă. Nu pierdeți parola, deoarece va trebui să o tastați ori de câte ori veți dori să folosiți această CA la semnarea de certificate pentru alții. Vi se vor cere și alte informații cum sunt numele țării (Country Name), numele comun (Common Name), etc. Rezultatul comenzii este stocat în două fișiere: `ca.key` și `ca.crt`. Fișierul `ca.key` conține cheia privata a CA, iar fișierul `ca.crt` conține certificatul cu cheia publică.

De asemenea, puteți specifica informațiile despre subiect și parola în linia de comandă, astfel încât nu vi se vor solicita informații suplimentare. În următoarea comandă, folosim `-subj` pentru a seta informația de subiect și folosim `-passout pass:dees` pentru a seta parola la `dees`.

```
openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 \
-keyout ca.key -out ca.crt \
-subj "/CN=www.modelCA.com/O=Model CA LTD./C=US" \
-passout pass:dees
```

Putem folosi următoarele comenzi pentru a analiza conținutul decodat al certificatului X509 și al cheii RSA (`-text` înseamnă decodarea conținutului în text simplu; `-noout` înseamnă a nu imprima conținutul versiunii codificate):

```
openssl x509 -in ca.crt -text -noout
openssl rsa -in ca.key -text -noout
```

Rulați comenzile de mai sus. Din rezultat, vă rugăm să identificați următoarele:

- Ce parte a certificatului indică că acesta este un certificat CA?
- Ce parte a certificatului indică că acesta este un certificat semnat de sine?
- În algoritmul RSA, avem un exponent public e, un exponent privat d, un modul n și două secrete, numerele p și q, astfel încât $n = pq$. Vă rugăm să identificați valorile pentru aceste elemente în certificatul dvs. și fișierele cheie.

2.2.2 Sarcina 2: Generarea unei cereri de certificat pentru serverul dvs.

O companie numită `bank32.com` (înlocuiește numele cu numele propriului server web) dorește să obțină un certificat de cheie de la CA noastră. Mai întâi trebuie să genereze o Cerere de semnare a certificatului (CSR), care practic include cheia publică și informațiile de identitate ale companiei. CSR-ul va fi trimis CA, care o va verifica informațiile de identitate din cerere și apoi va genera un certificat.

Comanda pentru a genera un CSR este destul de asemănătoare cu cea pe care am folosit-o la crearea certificatului semnat de sine al CA. Singura diferență este opțiunea `-x509`. Fără ea, comanda generează o cerere; cu ea, comanda generează un certificat autosemnat. Următoarea comandă generează un CSR pentru `www.bank32.com` (ar trebui să utilizați propriul nume de server):

```
openssl req -newkey rsa:2048 -sha256 \
            -keyout server.key -out server.csr \
            -subj "/CN=www.bank32.com/O=Bank32 Inc./C=US" \
            -passout pass:dees
```

Comanda va genera o pereche de chei publice/private și apoi va crea o cerere de semnare a certificatului din cheia publică. Putem folosi următoarea comandă pentru a examina conținutul decodat al CSR și fișiere cu cheie privată:

```
openssl req -in server.csr -text -noout
openssl rsa -in server.key -text -noout
```

Adăugarea de nume alternative. Multe site-uri web au adrese URL diferite. De exemplu, `www.example.com`, `example.com`, `example.net` și `example.org` indică toate către același server web. Datorită politicii de potrivire a numelui de gazdă impusă de browsere, numele comun dintr-un certificat trebuie să se potrivească cu numele de gazdă al serverului sau browserele vor refuza să comunice cu serverul.

Pentru a permite unui certificat să aibă mai multe nume, specificația X.509 definește extensii care trebuie anexate la un certificat. Această extensie se numește `Subject Alternative Name` (SAN). Folosind extensia SAN, este este posibil să se specifice mai multe nume de gazdă în câmpul `subjectAltName` al unui certificat.

Pentru a genera o cerere de semnare de certificat cu un astfel de câmp, putem pune toate informațiile necesare într-un fișier de configurare sau în linia de comandă. Vom folosi abordarea de linie de comandă în această sarcină (abordarea cu fișier de configurare este utilizată într-un alt laborator SEED, laboratorul TLS). Putem adăuga opțiunea la comanda "`openssl req`". Trebuie remarcat faptul că în câmpul de extensie `subjectAltName` trebuie să includeți pe cel din câmpul "`Commen Name`; în caz contrar, numele comun nu va fi acceptat ca nume valid.

```
-addext "subjectAltName = DNS:www.bank32.com, \
                          DNS:www.bank32A.com, \
                          DNS:www.bank32B.com"
```

2.2.3 Sarcina 3: Generarea unui certificat pentru serverul dvs.

Fișierul CSR trebuie să aibă semnătura CA pentru a forma un certificat. În lumea reală, fișierele CSR sunt de obicei trimise unei CA de încredere pentru semnătură. În acest laborator, vom folosi propria noastră CA de încredere pentru a genera certificate. Următoarea comandă transformă cererea de semnare a certificatului (`server.csr`) într-un certificat X509 (`server.crt`), folosind `CA.crt` și `ca.key`:

```
openssl ca -config myCA_openssl.cnf -policy policy_anything \
           -md sha256 -days 3650 \
           -in server.csr -out server.crt -batch \
           -cert ca.crt -keyfile ca.key
```

În comanda de mai sus, `myCA_openssl.cnf` este fișierul de configurare pe care l-am copiat din `/usr/lib/ssl/openssl.cnf` (am făcut și modificări acestui fișier în cadrul Sarcinii 2.2.1). Folosim `policy_anything` definită în fișierul de configurare. Aceasta nu este politica implicită; politica implicită are mai multe restricții, solicitând ca unele dintre informațiile subiectului din cerere să se potrivească cu cele din certificatul CA. Politica folosită în comandă, așa cum arată numele său, nu impune nicio regulă de potrivire.

Copiați câmpul de extensie. Din motive de securitate, setarea implicită din `openssl.cnf` nu permite comenzii "`openssl ca`" să copieze câmpul de extensie din cerere în certificatul final. Pentru a permite acest lucru, putem merge la copia noastră a fișierului de configurare și decomenta următoarea linie:

```
# Extension copying option: use with caution.
copy_extensions = copy
```

După semnarea certificatului, vă rugăm să utilizați următoarea comandă pentru a imprima conținutul decodat al certificatului certificat și verificați dacă sunt incluse numele alternative:

```
openssl x509 -in server.crt -text -noout
```

2.2.4 Sarcina 4: Plasarea unui certificat într-un sit de web HTTPS bazat pe Apache

Setarea serverului de HTTPS folosind comanda `openssl s_server` este folosită la depanare și demonstrații. În această lucrare, setăm un server de web HTTPS real, bazat pe Apache. Serverul Apache, care este deja instalat în VM, suportă protocolul HTTPS. Pentru a crea un sit de web HTTPS, trebuie doar să configurăm serverul Apache, astfel încât să știe de unde să ia cheia privată și certificatul. Dăm un exemplu de cum se activează HTTPS pentru un sit de web numit `www.example.com`. Sarcina voastră este să faceți același lucru pentru `SEEDPKILab2018.com` folosind certificatul generat în cadrul sarcinilor anterioare.

Un server Apache poate găzdui simultan mai multe situri de web. Are nevoie să știe directorul în care sunt stocate fișierele unui sit. Aceasta se face prin intermediul fișierului `VirtualHost`, aflat în directorul `/etc/apache2/sites-available`. În containerul nostru, avem un fișier numit `bank32_apache_ssl.conf`, care conține următoarea intrare:

```
<VirtualHost *:443>
    DocumentRoot /var/www/bank32
    ServerName www.bank32.com
    ServerAlias www.bank32A.com
    ServerAlias www.bank32B.com
    DirectoryIndex index.html
```

```
    SSLEngine On
    SSLCertificateFile /certs/bank32.crt     ①
    SSLCertificateKeyFile /certs/bank32.key ②
</VirtualHost>
```

Exemplul de mai sus configurează site-ul HTTPS `https://www.bank32.com` (portul `443` este implicit port HTTPS). Intrarea `ServerName` specifică numele site-ului web, în timp ce intrarea `DocumentRoot` specifică unde sunt stocate fișierele pentru site-ul web. Folosind intrările `ServerAlias`, permitem site-ul web să aibă nume diferite. Ar trebui să furnizați două intrări de alias.

De asemenea, trebuie să spunem lui Apache unde sunt stocate certificatul de server (linia ①) și cheia privată (linia ①). În `Dockerfile`, am inclus deja comenzile pentru a copia certificatul și cheia în directorul `/certs` al containerului.

Pentru a face site-ul să funcționeze, trebuie să activăm modulul `ssl` al Apache și apoi să activăm acest site. Acestea pot fi realizate folosind următoarele comenzi, care sunt deja executate la construirea containerului.

```
# a2enmod ssl // Enable the SSL module
# a2ensite bank32_apache_ssl // Enable the sites described in this file
```

Lansarea serverului Apache. Serverul Apache nu este pornit automat în container, din cauză că trebuie introdusă parola pentru a debloca cheia privată. Să mergem la container și să rulăm următoarele comandă pentru a porni serverul (enumerăm și câteva comenzi asociate):

```
// Start the server
# service apache2 start

// Stop the server
# service apache2 stop

// Restart a server
# service apache2 restart
```

Când pornește Apache, trebuie să încarce cheia privată pentru fiecare site HTTPS. Cheia noastră privată este criptată, așa că Apache ne va cere să introducem parola pentru decriptare. În interiorul containerului, parola folosită pentru `bank32` este `dees`. Odată ce totul este configurat corect, putem naviga pe site și tot traficul între browser și server vor fi criptate.

Vă rugăm să utilizați exemplul de mai sus ca ghid pentru a configura un server HTTPS pentru site-ul dvs. web. Descrieți pașii pe care i-ați făcut, conținutul pe care l-ați adăugat la fișierul de configurare Apache și capturile de ecran a rezultatului final care arată că puteți naviga cu succes pe site-ul HTTPS.

Director partajat între VM și container. În această sarcină, trebuie să copiem fișierele de pe VM în container. Pentru a evita recrearea în mod repetat a containerelor, am creat un director partajat între VM și container. Când utilizați fișierul `Compose` din directorul `Labsetup` pentru a crea containere, subdirectorul `volumes` va fi montat pe container. Tot ceea ce puneți în acest director va fi accesibil în interiorul containerului care rulează.

Navigarea pe site. Acum, navigați în browser la serverul dvs. web (notă: ar trebui să puneți `https` la începutul URL-ului, în loc să utilizați `http`). Vă rugăm să descrieți și să explicați observațiile dvs. Cel mai probabil, nu vei putea reuși, asta pentru că... (motivele sunt omise aici; trebuie să

dați o explicație în raportul de laborator). Vă rugăm să remediați problema și să demonstrați că puteți vizita cu succes situl de web HTTPS.

În cele ce urmează, oferim instrucțiuni despre cum să încărcați un certificat în Firefox. Intenționat nu explicăm de ce și ce certificat ar trebui încărcat; trebuie să vă dați seama singuri. Pentru a adăuga manual un certificat în browserul Firefox, introduceți următorul URL în bara de adrese și faceți clic pe butonul `View Certificates` de pe pagină (defilați în jos).

```
about:preferences#privacy
```

În tab-ul `Authorities`, veți vedea o listă de certificate care sunt deja acceptate de Firefox. De aici, putem importa propriile certificate. După ce ați ales fișierul de certificat, vă rugăm să selectați următoarea opțiune: "Trust this CA to identify web sites". Veți vedea că certificatul nostru este acum în lista de certificate acceptate a lui Firefox

2.2.5 Sarcina 5: Lansarea unui atac de tipul om-la-mijloc

În cadrul acestei sarcini, vom arăta cum PKI poate înfrânge atacurile de tipul om-la-mijloc (engl. Man-In-The-Middle (MITM)). Figura 1 ilustrează cum funcționează atacurile de tipul MITM. Să presupunem că Alice dorește să viziteze `example.com` folosind protocolul HTTPS. Ea are nevoie să obțină cheia publică de la serverul `example.com`; Alice va genera un secret și va cifra secretul folosind cheia publică a serverului și o va trimite la server. Dacă un atacator poate intercepta comunicarea între Alice și server, atacatorul poate înlocui cheia publică a serverului cu propria cheie publică. Din acest motiv, secretul lui Alice is cifrat de fapt cu cheia publică a atacatorului și atacatorul va putea citi secretul lui Alice. Secretul este folosit la cifrarea comunicației dintre Alice și server, așa că atacatorul poate descifra comunicația.

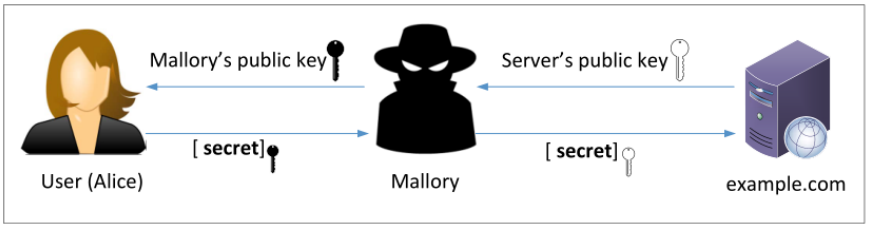

Figura 1: Cum funcționează atacul MITM

Scopul acestei sarcini este să ajute la înțelegerea modului în care PKI poate înfrânge asemenea atacuri. În cadrul sarcinii vom emula un atac MITM și vom vedea cum poate PKI să-l combată. Mai întâi vom alege un sit de web drept țintă. În această lucrare vom folosi `example.com` ca țintă, dar dvs. puteți alege un sit popular, cum ar fi situl unei bănci sau al unei rețele sociale.

Pasul 1. Setarea sitului rău intenționat. În cadrul sarcinii 2.2.4, am setat deja un sit de web HTTPS. Vom folosi același sit de web pentru a ne da drept `example.com` (sau un alt sit ales de Dvs.). Pentru a realiza acest lucru, vom folosi instrucțiunile din sarcina 2.2.4 pentru a adăuga o intrare `VirtualHost` în fișierul de configurare a SSL pentru Apache: `ServerName` ar trebui să fie `www.example.com`, dar restul configurației poate fi identică cu cea din sarcina 2.2.4.

Scopul nostru este următorul: când un utilizator încearcă să viziteze `www.example.com`, vom face ca utilizatorul să aterizeze în situl nostru, care găzduiește un sit de web fals pentru situl `www.example.com`. Dacă acesta ar fi fost un sit de rețea socială, situl fals ar trebui să afișeze o pagină de login similară cu cea a sitului țintă. Dacă utilizatorii nu pot observa diferențele, atunci ei ar putea să-și tasteze acreditările (credențialele) contului în pagina falsă, dezvăluindu-le astfel atacatorului.

Pasul 2. Devenim om-la-mijloc. Există câteva moduri în care putem face ca cererile de HTTPS ale utilizatorului să aterizeze în serverul nostru. Una dintre acestea este să atacăm rutarea, astfel încât cererea de HTTPS a utilizatorului să fie rutată spre serverul nostru.

O altă cale este să atacăm DNS, astfel ca atunci când mașina victimei încearcă să afle adresa IP a serverului țintă, să primească adresa serverului nostru. Pentru această sarcină vom simula atacarea DNS. În loc să lansăm un atac real pentru otrăvirea cache DNS, vom modifica pur și simplu fișierul `/etc/hosts` al mașinii victimei pentru a emula rezultatul otrăvirii cache DNS prin maparea gazdei `www.example.com` la adresa serverului rău intenționat.

```
10.9.0.80 www.example.com
```

Pasul 3. Răsfoirea sitului țintă. Cu toate setate, vizitați situl țintă real și vedeți ce va spune browserul.

Explicați ce ați observat.

2.2.6 Sarcina 6: Lansarea unui atac de tipul om-la-mijloc cu o CA compromisă

În această sarcină, presupunem că CA rădăcină creată în Sarcina 2.2.1 este compromisă de un atacator și cheia sa privată este furată. Prin urmare, atacatorul poate genera orice certificat arbitrar folosind cheia privată a acestei CA. In această sarcină, vom vedea consecințele unui astfel de compromis.

Proiectați un experiment pentru a dovedi că atacatorul poate lansa cu suces atacuri MITM asupra oricărui sit HTTPS. Puteți folosi aceeași setare ca cea din sarcina 2.2.5, dar, de această dată, trebuie să demonstrați că atacul are succes, adică browser-ul nu va avertiza atunci când victima încearcă să viziteze un sit de web, dar ajunge în situl de web fals al atacatorului MITM.

3 Trimiterea rezultatelor

Trebuie să trimiteți un raport detaliat în care să descrieți ce ați făcut și ce ați observat; de asemenea trebuie să explicați observațiile surprinzătoare sau interesante. În cadrul raportului trebuie să răspundeți la toate întrebările puse în această lucrare de laborator.