A TLS Stack

PEOPLE
APPLICATIONS
LIBRARIES
PROTOCOLS
CRYPTO
MATH

You are here
Disclaimer

- This hands-on session is focused on security aspects, not
  - Accessibility
  - Performance
  - Portability
  - Efficient maintenance

- Not even all known security aspects are covered
Structure

- Today
  - Act I: “Let’s get a padlock on our web site!”

- In the future
  - Act II: “Configure TLS well on the server!”
  - Act III: “Make the web app play nice with TLS!”
- Linux: Fedora 20 ([link](http://fedoraproject.org/en/get-fedora))
  - Host name: learn.tls.now
  - User “secappdev”, password “secret”
  - We will use the command line:
    
```bash
$ user_command   <- type as 'user_command'
# root_command   <- same as 'sudo root_command'
```
  - We will edit configuration files
Already installed: Apache web server 2.4.6

- `# yum install httpd`
- `# systemctl enable httpd.service`
  - The document root is in `/var/www/html/`, linked from desktop
  - The configuration directory is `/etc/httpd`

Already installed, but inactive: mod_ssl

- `# yum install mod_ssl`

Already installed: OpenSSL 1.0.1e

- `# yum install openssl`
ACT I

“Let’s get a padlock on our web site!”
Act I Completion Criteria

- http://learn.tls.now loads without problems
- https://learn.tls.now loads without problems
Try It!

HTTP woks out of the box

HTTPS doesn’t work yet
Wireshark

- Launch Wireshark
- Browse the site through http
- Find the password in the traffic
Act I Completion Criteria (2)

- ✔️ http://learn.tls.now loads without problems
- ❌ https://learn.tls.now loads without problems
  - ❑ No connection error
Enable “https”

- "Activate" mod_ssl:
  
  ```bash
  # mv /etc/httpd/conf.d/ssl.conf.dead
  /etc/httpd/conf.d/ssl.conf
  ```

- Tell Apache to reload its configuration
  
  ```bash
  # systemctl reload httpd.service
  ```

- Refresh the browser...
Try it again!

- If the user continued:
  - Confidentiality OK
  - Integrity OK
  - Authenticity of the server UNKNOWN

- If your production site is like this you’re teaching your users bad habits!
  - And you are hurting our collective security!

STOP
"Technical Details" of Firefox Warning

- “This Connection is Untrusted”
- ...
- Technical Details
  - learn.tls.now uses an invalid security certificate.
  - The certificate is not trusted because it is self-signed.
Security Certificate?

- I did not do anything about certificates!
- You did not do anything, either!
- But Fedora did:
  - The `mod_ssl` installation generated a dummy certificate
  - Why: it gets the server up and running without friction
    - Good for *novices*, they can learn later to do it properly
    - Neutral for *professionals*, they are supposed to have a secure procedure anyway
- Most Linux distros do it this way
Not Trusted?

- “because it is self-signed”
  - Misnomer: the trusted Root CA certificates are self-signed, too
  - Should be: “… because certificate signer is not trusted.”

- For continuing the browser asks the user: “Do you trust the certificate signer?”
  - When continuing, the user answers “Yes”
  - Actually, most users just want to answer “Get out of my way!”
  - Risky - the certificate could belong to an attacker-in-the-middle!
  - The “Add exception… permanently” option makes the risk permanent

- Lesson: avoid exposing your users to this warning in the first place!
Act I Completion Criteria (3)

- ✓ http://learn.tls.now loads without problems
- ✗ https://learn.tls.now loads without problems
  - ✓ No connection error
  - ✗ No trust warning
Where is the Certificate?

- You have already met `/etc/httpd/conf.d/ssl.conf`
- In it you can find the directives:
  
  ```
  # TLS Configuration per virtual host
  <VirtualHost _default_:443>
      SSLEngine on
      SSLCertificateFile /etc/pki/tls/certs/localhost.crt
      SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
  ...
  </VirtualHost>
  ```
In Fedora 20 the version of OpenSSL is 1.0.1e

OpenSSL has a command mode:

```
# openssl SUBCOMMAND OPTIONS
```

Which subcommands?

- Run `man openssl`, see the COMMAND SUMMARY section
- Then run `man SUBCOMMAND`

Options are in single dash format

- E.g.: `-noout`, “do not re-display raw input”

Some options can have parameters

- E.g.: `-in INPUTFILE`, “read input from INPUTFILE, not stdin”
Inspect Certificate with OpenSSL (1)

```
# cd /etc/pki/tls/
# openssl x509 -in certs/localhost.crt -text -noout
Certificate:
  Data:[...]
  Signature Algorithm: sha1WithRSAEncryption
  Issuer: C=--, ST=SomeState, [...] 
  Validity
    Not Before: Feb 11 09:35:21 2014 GMT
    Not After : Feb 11 09:35:21 2015 GMT
  Subject: C=--, ST=SomeState, [...] 
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    Public-Key: (1024 bit)
    Modulus:[...]
    Exponent: 65537 (0x10001)
```
Inspect Certificate with OpenSSL (2)

(continued)

X509v3 extensions:

X509v3 Basic Constraints:

CA: FALSE

X509v3 Key Usage:

Digital Signature, Non Repudiation, Key Encipherment

Signature Algorithm: sha1WithRSAEncryption
Act I Completion Criteria (4)

- http://learn.tls.now loads without problems
- https://learn.tls.now loads without problems
  - ✓ No connection error
  - ✗ No trust warning
    - ✗ Private key of at least 2048 bits
    - ✗ Public exponent of at least 17 bits
    - ✗ Certificate corresponding to the domain
    - ✓ Certificate corresponding to private key
    - ✓ Signature on certificate valid
    - ✗ Certificate signing key is trusted (maybe through recursion)
    - ✓ Signature algorithm not using MD5
    - ✓ Validity period of 1 year
    - ✓ Usage compatible with being a web site certificate

...
We Need to Generate a New Certificate

Steps:
1. You generate a key pair
2. You generate a Certificate Signing Request (CSR)
3. A Registration Authority verifies the CSR
4. A Certificate Authority creates the certificate for you

There are numerous helper tools and wizards
- Fedora recommends `genkey` from package `crypto-utils`
- Fedora provides OpenSSL GUI package `xca`
- Fedora provides heavyweight set of packages `pki-*`
- OpenSSL wrapper: `/etc/pki/tls/misc/CA`
- OpenSSL subcommand: `openssl ca`

We will take the route that shows a lot of detail
Generate a Keypair

- Keys will be for RSA (a little simpler than for DSA)
- Default keyfile format is PEM, accepted by Apache
  - The conventional keypair file extension is `.key`
- Key length of at least 2048 bits is based on estimated strength
  - NIST Special Publication 800-57 (July 2012)
  - CA/Browser Forum Baseline Requirements (November 2011)
- Public exponent of at least 32 bits, because of:
  - 2006: Belichenbacher attack against exponent “3” PKCS #1 v1.5
  - SecAppDev 2012, Bart Preneel: “use >32 bits”
- So far, recommended key generation:
  ```
  # openssl genpkey -algorithm rsa -out private/keys.key
  -pkeyopt rsa_keygen_bits:2048
  -pkeyopt rsa_keygen_pubexp:0x100000001
  ```
Keyfile Protection

- **Option 1**: password protect the keyfile
  - Add cipher option, e.g. `aes-128-cbc`
  - By default, passphrase provided interactively at generation
    - See `man openssl`, section PASS PHRASE ARGUMENTS
  - But passphrase must be provided every time Apache starts up
    - See docs for SSLPassPhraseDialog

- **Option 2**: keyfile protected at filesystem and SELinux level
  
  ```
  # chmod u=rw,go= private/keys.key
  # selinux...
  ```
Act I Completion Criteria (5)

☑️ http://learn.tls.now loads without problems
☒️ https://learn.tls.now loads without problems

☑️ No connection error
☒️ No trust warning

☑️ Private key of at least 2048 bits
☑️ Public exponent of at least 17 bits
☐ Certificate corresponding to the domain
☐ Certificate corresponding to private key
☐ Signature on certificate valid
☐ Certificate signing key is trusted (maybe through recursion)
☐ Signature algorithm not using MD5
☐ Validity period of 1 year

…
The Certificate Signing Request

- Command:
  
  ```bash
  # openssl req -new [OPTIONS] 
  -key private/keys.key -out certreq.csr
  ```

- One-year validity period with `-days 365`

- Default hash algorithm is SHA1, not MD5

- Other constraints do not have command line options, but are indicated in a configuration file
OpenSSL Configuration Files

- Config file can be default or specified with `-config CONFIGFILE`
  - On Fedora the default file is: `/etc/pki/tls/openssl.cnf`
- Sections are introduced by lines like `[ section_name ]`
- Section `[ v3_req ]` has some good values:
  - `basicConstraints = CA:FALSE`
    - If lost, the private key can not be used to sign other certificates
  - `keyUsage = nonRepudiation, digitalSignature, keyEncipherment`
    - Compatible with being a web site certificate
- Use that section with `-extensions v3_req`
The Domain Name for the Certificate

- Express the domain name as a Distinguished Name: 
  `-subject
    
    /C=BE/ST=Vlaams Brabant/L=Leuven/O=SecAppDev/CN=learn.tls.now/
  `

- What about co-hosting related web sites on the same IP address?
  - https://learn.tls.now:443, but also
  - https://www.tls.now:443 and
  - https://tls.now:443

- Same web server, same configuration, cannot serve more than one certificate...

- Specify all names in the `[v3_req]` section of the configuration file, which the simulated CA will later use:
  ```
  subjectAltName=DNS:learn.tls.now,DNS:www.tls.now,DNS:tls.now
  ```
CSR Generation Summary

- Generate a CSR (Certificate Signing Request)
  
  # openssl req -new -subj "/C=BE/ST=Vlaams Brabant/L=Leuven/O=SecAppDev/CN=learn.tls.now/"
  -days 365 -extensions v3_req
  -key /etc/pki/tls/private/keys.key
  -out /etc/pki/tls/certreq.csr
Anyone can generate a CSR for your web site…
You need to be able to prove to strangers it’s your web site
Employ Trusted Third Parties
   – professional Registration Authorities
   – professional Certification Authorities
Luckily, browsers come preloaded with trusting a bunch of them
   – Or “Unfortunately,…” (in case one of them is corrupted)
Certifying your public key usually has a price tag
We will simulate the CA’s procedure
Extended Validation Certificates

- What exactly is certified?
- It should be: “requester’s ownership of the FQDN”
- How stringent checks?
  - Verify photo ID of the requester?
  - Verify business ownership of the FQDN?
  - Verify control of the indicated e-mail address?
  - Verify possession of the private key?
  - Don’t verify, just issue a limited-time “testing” cert?
- “Extended Validation” certificates
  - a.k.a. “the way it should have been done in the first place”
  - Validation requirements are no less than a known threshold
  - Certificates contain a reference to the CA’s verification policy
- Current browsers display additional positive assurance for EV certs
Simulated CA Step

- Sign the Certificate Signing Request
  - Prepare a file to keep track of the serial numbers
  - Prepare a file to keep track of the issued certificates
  - Configure various other values in `openssl.conf`
    
    ```
    $ ~/ca/casign /etc/pki/tls/certreq.csr
    ```

- The passphrase for the CA’s private key is “trustedca”

- The signed certificate will be saved to `/etc/pki/tls/certs/casigned.crt`
Deploy Certificates to Apache

- The files are already in the folders expected by Apache on Fedora
  - Therefore they have inherited the correct SELinux context
- Let `mod_ssl` know about them
  - Replace old values in `/etc/httpd/conf.d/ssl.conf`:
    ```
    SSLCertificateFile /etc/pki/tls/certs/casigned.crt
    SSLCertificateKeyFile /etc/pki/tls/private/keys.key
    ```
- Let Apache know about the change
  ```
  # systemctl reload httpd.service
  ```
  - Use `systemctl reload`, because we just changed configs
It Worked!

Welcome!

(User "secappdev", password "secret")

User name: [Field]
Password: [Field]
Act I Completion Criteria (6)

☑️ http://learn.tls.now loads without problems
☑️ https://learn.tls.now loads without problems
    ☑️ No connection error
    ☑️ No trust warning
        ☑️ Private key of at least 2048 bits
        ☑️ Public exponent of at least 17 bits
        ☑️ Certificate corresponding to the domain
        ☑️ Certificate corresponding to private key
        ☑️ Signature on certificate valid
        ☑️ Certificate signing key is trusted (maybe through recursion)
        ☑️ Signature algorithm not using MD5
        ☑️ Validity period of 1 year

…
Discussion: Certificate Chains

- There are sub-CAs and sub-sub-CAs, …
- Browser only knows root CA certificate in the beginning
- Web site must provide the intermediate links in order:
  - TLS RFC says so: each certificate certifies the one before it

```
-----BEGIN CERTIFICATE-----
MIICzzCCAbegAwIBAgIJAIXRQJSKK2HAYD
Ap Sub-sub-CA certificate FLVQOD
U2olbGhvc3QubG9jYWxkb
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIDCTCCAnKgAwIBAgICNtQwDQYJKoDB
MR Sub-CA certificate ETb2Nh
IGS25sSHgrz8OU28eA==
-----END CERTIFICATE-----
```

- The cert chain goes into the file specified in ssl.conf
  SSLCertificateChainFile
  /etc/pki/tls/certs/server-chain.crt
Discussion: Certificate Expiration and Revocation

- At expiry, assume private key to be compromised
  - Cryptanalysis: deriving it from the public key
  - Breach: breaking into web server and copying the file
  - Brute force 1: trying enough candidates for a match
  - Brute force 2: “convince” a sysadmin to disclose it

- Generate a new keypair for every new certificate, it’s cheap
- Reasons for renewing the keypair even before expiration:
  - You have evidence of key compromise (see above)
  - You have lost control of the private key (no disaster recovery)

- Inform the CA, who updates a CRL or an OCSP server
- Browsers would ignore the old cert if they encountered it
Bibliography

- **Fedora 18 System Administrator’s Guide, Ch. 14, Web Servers:**

- **Apache 2.4 Module mod_ssl:**
  http://httpd.apache.org/docs/2.4/mod/mod_ssl.htm

- **OpenSSL man pages and HOWTOs:**
  http://www.openssl.org/docs/

- **OWASP Transport Layer Protection Cheat Sheet:**
  https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet
QUESTIONS?

SEE YOU AGAIN AT SECAPPDEV 2014!