Technische Universität München
Lehrstuhl Informatik VIII
Prof. Dr.-Ing. Georg Carle
Dr. Heiko Niedermayer
Cornelius Diekmann, M.Sc.
Dr. Ralph Holz

Network Security Winter Term 2014/2015
Exercise 3

Task 1 The X.509 PKI for the WWW

Recall that certificates are used to establish a cryptographically secure binding between an identity and the public key belonging to that entity: $Cert_{CA}(B) = Sig_{CA}(B, K_{\text{B-pub}})$. In this task, we will have a closer look at the Public Key Infrastructure (PKI) as defined in the X.509 standard. Today, the most important use case for X.509 certificates is to secure HTTP connections with SSL/TLS. Here, certificates are issued to web sites and browsers rely on them to establish secure connections. We will begin with a few theoretical questions and then turn to practically evaluating X.509 certificates in this task. The following RFC defines X.509: `http://www.ietf.org/rfc/rfc5280.txt`.
Figure 1 shows a simplified X.509 PKI. The Root CA does not issue certificates to end-entities itself. Rather, it delegates this task to Intermediate CAs.

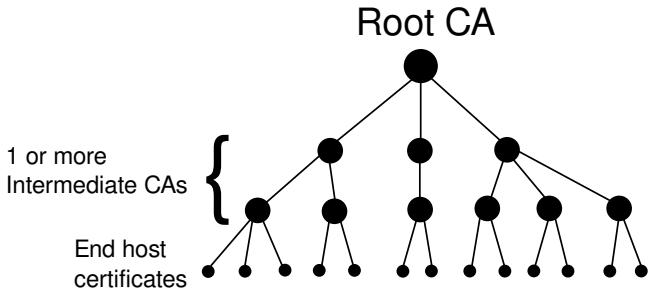

Figure 1: Ideal X.509 PKI with one Root CA.

You may disregard anything that would be stored as an X509v3 extension (see lecture slides) for the first tasks.

a) Assume Alice has been issued a certificate. Let Alice be the leftmost leaf in Figure 1. Bob wishes to verify that a given public key belongs to Alice. He has obtained her certificate and trusts the root certificate. How does he proceed to check whether the certificate is valid (pay attention to how he finds the path!)? Concerning checks on certificate fields, you may restrict yourself to the 'vital' certificate fields (see slides).

b) When Bob goes through the steps, which entities must he implicitly 'trust' to issue correct certificates?

For the following tasks, you need to download the archive from the course website.
Change to the directory where the `ca-certificates.crt` file is located.

c) One use of X.509 today is to allow SSL/TLS-secured connections to WWW servers. Let us have a

look at the certificates of a few hosts. Use a Unix/Linux machine for the following. Open a terminal and issue the following commands:

- `openssl s_client -connect twitter.com:443`
- `openssl s_client -connect twitter.com:443 -CAfile ca-certificates.crt`

Now answer the following questions:

1. How do the two commands behave differently? Note: If they do not behave differently, replace the first command with `openssl s_client -connect twitter.com:443 -CApath /dev/null`. If the results are still the same, your OS is horrible.
2. Thus, what does the `verify error` you get in the first command indicate?
3. Is there an *intermediate certificate*? If yes, what purpose does it serve?
4. Is there an *intermediate CA*, i.e. is there more than one organization/company involved in the certification? Say why you think so.

d) Copy the part beginning with `---BEGIN CERTIFICATE---` up to `---END CERTIFICATE---` into a file `twitter.com.cert`.
If the command in the previous task did not work, you can use the first certificate in `twitter.com_chain.pem`. Then issue the following command: `openssl x509 -in twitter.com.cert -text`. The result is a text representation of the certificate content. Explain what is stored in an X.509 certificate (i.e. say what is stored in each field). You may skip all X.509v3 extensions except `CRL Distribution Points`, `Authority Information Access` and `Subject Alternative Name`.

e) Implement certificate verification.

1. From the downloaded archive, complete the provided template (`main.py`).
2. What is the role of the `ca-certificates.crt` file?
3. Which of the provided certificates and certificate chains (see list below) is valid?
4. What is the problem for the invalid certificates?

- `ahmed_root.pem`
- `expired_cert.pem`
- `twitter_cert.pem`
- `twitter.com_chain2.pem`
- `twitter.com_chain.pem`
- `www.twitter.com_by_ahmed.pem`

f) Browse through the file `ca-certificates.crt`.

1. How many certificates are in there? Give the command you have used to count.

2. Browse a bit more and try to find 2 Root CAs that you wouldn't have expected and/or find suspicious. Explain why you think they're unexpected or suspicious.

g) Now assume that a Root CA in the root store is hacked and under the control of an attacker, and this is not noticed by anyone for a few months. (Note: this question is intentionally a bit more open, we want you to think and discuss)

1. What further attacks can the attacker stage now? Sketch a possible attack setup.

2. In the attack you have described above, can browser users rely on CRLs or OCSP to protect them? Why?