Assignment Four

Protecting the proxy network

Set: *6th of June 2011*
Due: *17th of June 2011 @ 23:55 CEST*

Synopsis:
Implement authentication and transmission encryption for the distributed anonymizing proxy network.

Introduction

This is the last of the four assignments in the *Datanet* course. The four assignments are practical in nature, and you will gradually build a distributed anonymising web proxy. Your proxy will be part of a real distributed system running on the Internet, with peers contributed by your fellow students. This fourth assignment is about ensuring the validity of requests, and protecting the communication in the proxy network from tampering and peeking.

In this assignment you must write a peer, that supports request verification, response signing, and encrypted communication, for the distributed proxy network described in this assignment and Assignment 3. You may choose to extend your implementation from Assignment 3.

Implementation

Your implementation will require that you use various cryptographic services. You are not required to implement these services yourself (though you can if you like) and you may instead use a suitable library. If you are using Python you could, for example, use the *PyCrypto*[1] or *tlslite*[2] libraries. For other languages you will have to find some other suitable crypto library. Most languages should at least provide an interface to the *OpenSSL*[3] library.

You should keep in mind that the functionality that you need out of your crypto library may be at a lower level than what it actually provides. For example, to use the *tlslite* library you will have to forego its convenience implementations of `sign` and `verify`, as these use a data format that is incompatible with that used in this assignment. It is however possible, by inspecting how these methods are implemented, to make your own `sign` and `verify` methods that work with the proxy network.

[1] `http://www.dlitz.net/software/pycrypto/`
[2] `http://trevp.net/tlslite/`
[3] `http://www.openssl.org/`

Public/Private Key Pair

You will need to generate an asymmetric RSA keypair that is used in the proxy network. The public part must be reported to the tracker and the private part is used for signing and encrypting in your peer. The generated key must be a 1024 bit RSA key and how you store it, internally to your application, is up to you. You will have to send the key to the tracker in the format specified in its documentation: `http://datanet2011tracker.appspot.com/doc/pubkeys`.

Tracker Communication

The tracker documentation, `http://datanet2011tracker.appspot.com/doc/`, specifies how your peer must interact with the tracker in order to enable its cryptographic services:

- Your peer *should* verify the signature of the response returned by the tracker[4], using the trackers public key[5].
- For your peer to receive public keys for other peers, your peer must sign its requests to the tracker. An unsigned request will **never** return any public keys.
- For other peers to see your trackers public key, your tracker must submit its public key as part of its registration with the tracker.

For exact information on the protocol used to interact with the tracker, you should refer to the tracker documentation.

Peer Communication

If you are communicating with a peer which has a public key, you should communicate with that peer using an encrypted exchange. The method of encryption used between peers is given below:

- For each request, create a random session key (AES-256 + IV)
- Encrypt the HTTP request with the session key (CBC, PCKS7 padding)
- Encrypt the session key with the destination proxy's public key (RSA-1024)
- Create an envelope request with an encrypted payload

[4]When communicating with a peer that has signed its request to the tracker, a third party may from time to time try to alter the data sent in a response. This can be detected by verifying the response from the tracker.

[5]The tracker's public key can be found on the tracker's home page.

Encrypted Request Envelope

The format of the envelope request is as follows:

```
DATANET * HTTP/1.1
Host: *
Session-Key: <encrypted session key>
Content-Length: <size of encrypted payload>
<CR><LF><CR><LF>
... encrypted payload ...
```

This envelope ensures that all HTTP capable transport mechanisms will correctly be able to forward (but not cache) the message, while still introducing encryption to the transmission.

Since AES is used in CBC mode it is more secure if the Initialization Vector (IV) is randomly chosen for each request. The blocksize of AES-256 is 128 bits, with a keysize of 256 bits, which makes the session key $16 + 32 = 48$ bytes long. This resulting 48 byte array must then be encrypted using the RSA-1024 public key, resulting in a 128 bytes encrypted value. As this value is binary, it must then be encoded with base64 encoding to fit in the http header. After the base64 encoding, the resulting string should be 172 characters.

Decrypting the Request

To correctly handle an encrypted request, your proxy must do the following:

- Detect the `DATANET` HTTP method
- Read the payload and encrypted session key
- Decrypt the session key with the proxy private key
- Decrypt the payload with the session key
- Either forward the message to another proxy (wrapped in a new envelope)
- Or forward the request to the remote destination

Encrypted Responses

When the proxy receives a response it must also respond with an encrypted payload, **if** the request was encrypted. This ensures that there is always a known session key, but also that the initial request (from the browser) is unencrypted. The encrypted response must be wrapped in a header using the following format:

```
HTTP/1.1 700 Encrypted
Cache-Control: no-cache
Content-Length: <size of encrypted payload>
<CR><LF><CR><LF>
... encrypted payload ...
```

Experiment

You must also conduct an experiment that uses the proxy network, where you investigate the latency and bandwidth of the proxy network now that you have introduced encryption. You should use your timings from the previous assignment for comparison (or generate some new timings without using encryption). You should conduct *at least* the following experiments:

- Start a peer and make your browser use the peer as its proxy. Load some of the same pages as you did in Assignment 2 and measure the latency. Beware that the nature of the proxy network will make your timings vary and you will have to run the test a few times to get an average.

  You *may* want to force your proxy to set the `Via` header, for requests that come from your browser, so that you can correlate the route with your timings.

- Start a peer and make your browser use the peer as a its proxy. Download a large file (10-100MB) and measure the bandwidth. You will have to run the test a few times to get an average.

Your Report

Your report *MUST* be written in ACM format. An ACM template for LaTeX and MS-Word is available for download via Absalon.

Your reports should contain:

- An abstract describing the contents of your report
- A description of the encryption extensions
- A description of the threat (what does the system protected against?)
- A description of how the extensions improve the system security
- A description of any flaws or problems with the security system (where can you attack it?)
- A description of the types of anonymity offered by the systems (what is (not) anonymized?)
- The responsiveness and bandwidth measures that you obtained from your experiment

Deliverables for This Assignment

You are encouraged to work in informal groups for this assignment, for the purpose of discussing implementation details and limitations. We strongly encouraged you to come to the exercise, where we will use time to discuss the design, implementation etc. The implementation and report that you hand in must however be **your own individual work**. You should submit the following items:

- A single PDF file, A4 size, no more than 3 pages, in ACM format, describing each item from report section above
- A single ZIP/tbz2 file with all code relevant to the implementation

Handing In Your Assignment

You will be handing this assignment in using Absalon. Try not to hand in your files at the very last minute, in case your private key is stolen by a SPECTRE operative leaving you with no way to sign your assignment. **Do not email us your assignments unless we expressly ask you to do so**.

Assessment

You will get written qualitative feedback, but no grade for the assignment. Your assignments will be evaluated together with your exam and produce a final grade for the course.