LTH EITN50 Advanced Computer Security

Project 2 - Object Security

adsec03

Stefan Eng: atn08sen, Rasmus Olofzon: muh11rol

2017-09-19

Introduction

This report is written as a complement to source code developed as a solution to an assignment in the course Advanced Computer Security (EITN50) at the Faculty of Engineering at Lund University. The assignment was to imagine a hypothetical situation where one works at a company as a senior security engineer and is assigned with the task of developing a proof-of-concept implementation that utilises the concepts of object security.

Architecture

Our solution consists of one file, objectsec.py. It contains both the server and the client logic, specified by running it with different flags (server: 'python3 objectsec.py –server', client: 'python3 objectsec.py –client').

The server logic lies in the method `server()`. Basically it enters a loop where it awaits arrival of commands to parse (done in inner method `parse_commands()`).

The client logic lies in the method `client()`. It performs a handshake and then sends a couple of different messages, tabulated below:

Message index	MAC	Data	Encryption	Sequence nbrs
0	OK	OK	OK	OK
1	Poisoned	OK	OK	OK
2	OK	Poisoned	OK	OK
3	OK	OK	Poisoned	OK
4	OK	OK	OK	Jump
5	OK	OK	OK	Too low
6	OK	OK	OK	Large jump

Above these two client and server methods there are several utility methods. These include:

- `def encrypt(key, message)`
  Encrypts a message with AES.
- `def decrypt(key, message)`
  Decrypts a message with AES.
- `def get_prime()`
  Calculates prime numbers using Erathostenes sieve, in `def super_secret_primes()`.
- `def calculate_shared_secret(computed_primitive, own_secret, public_mod)`
  Calculates a shared secret.
- `def generate_mac(key, message)`
  Generates MAC with PyCrypto class Hash.HMAC.
- `def send_data(socket, data, address, key="", poison="", poison_message=False)`
  This is used for sending data. The parameters `poison` and `poison_message` is used for tampering with messages before sending, as described above. The method increases sequence number, calculates MAC, encrypts the data and sends the data through the socket.
- `def get_data(socket, key="")`
  This is used for receiving data, decrypting it, tokenising it and checking integrity through calculating MAC.

Requirements

The solution should:

1. work on the principle of object security

The security is provided in the application layer, here the client and server-structure written in Python.

2. provide integrity, confidentiality, and replay protection

Integrity is provided through a MAC, which is computed with PyCrypto class Hash.HMAC.

Confidentiality is provided through encryption, which is done with PyCrypto module Cipher.AES.

Replay protection is provided through sequence numbers, which are started from 0 and then incremented. The rationale for this was that the messages are both encrypted and a MAC is calculated, as well as the fact that this was to be a proof-of-concept implementation, resulting in a trivial implementation of sequence numbers.

3. use UDP as the way to exchange data between the two parties

We use the socket python module and when socket object(s) is(are) created the type parameter is specified to socket.SOCK_DGRAM. Hence, UDP is used.

4. work on the principle of forward security

The solution for this was to use ephemeral keys, i. e. keys that are generated for each session.

5. should have at least two distinct parts; handshake and (protected) data exchange

The two parts are: the handshake methods `handshake(socket, address)` (server) and `handshake(address)` (client) and the `send_data(socket, data, address, key="", poison="", poison_message=False)` and `get_data(socket, key="")` methods.

6. actually work when we test it

The objectsec.py file is included in the mail containing this report. Our solution is also hosted on https://github.com/Wribbe/50ietn-objectsec, the repository is cloneable when using the http protocol. Our solution uses PyCrypto, either use the Crypto folder in the repo or delete that folder and use your own installed version. PyCrypto is here considered to be a default library/component, as per the assignment wording [1]. We hope that there will be no problems executing our solution.

7. document the design choices for your implementation

See Architecture section above.

Conclusion

The result is a simple but functional proof of concept, and we hope that our hypothetical audience at the company will be satisfied with our solution.

[1] http://www.eit.lth.se/fileadmin/eit/courses/eitn50/Project_ObjSec/Project_OBJsec.pdf , Project description to which this solution was developed. Retrieved 2017-09-19