Resilient Networks - Exercise 4

Luca Gebhardt, Jan Pfeifer

January 17, 2024

1 DoS Evaluation

1.1 Sensor Communication

1. A possible attack vector could be to send many random messages to the server which then has to decrypt and validate the signature ($> 100ms$). Since it takes only $< 15ms$ to encrypt and sign the messages for the attacker the server has to use way more resources in comparison. By flooding the server it is going to have to take a lot of time and resources which may lead to the depletion of computing power.

2. The computing power would affected and depleted.

3. The fundamental vulnerability is the long duration the "brute force protection" takes. This attack could be avoided if we would limit the rate to users which cost a lot of resources.

1.2 Data Visualization

1. We notice that ID's can be requested multiple times in one request and the size of the returned data increase when more ID's are requested.

2. By requesting sensor ID 1 multiple times in one request, we can generate a lot of traffic which leaves less bandwidth for actual users.

3. The attacker could spoof the IP address of his victim and send a request that contains e.g. the ID 1 multiple times. Due to the increased response compared to the request, it is an amplification. According to the description, the API performs a source address validation. So, the API should not be usable in an amplifier attack that uses a spoofed IP address

2 Client Puzzle

2.1 Goal and Concept

1. A Client Puzzle protects against CPU exhaustion attacks in which an attacker opens multiple connections to the victim server. For every new connection the attacker has to solve a problem, that introduces a delay on the attacking side. Normal users would just notice a slight delay, but because the attacker opens multiple connections at once and has to solve a Client Puzzle for each connection, the computational resource is depleted on the attackers side, leading to a reduction in the efficiency of the attack.

2. The fundamental vulnerability lies in a inequality between the resources that a client has to use to send a request and the resources the server has to use to generate an answer. The goal of a Client Puzzle is to equalize this situation and balance the computational power used at client and server side.

3. There are cryptographic primites like hashing, signing or asymmetric encryption algorithms which are hard to calculate but easy to verify. The puzzles' difficulty can be easily scaled based on factors such as server load or server trust of the client by adjusting the length of the data that has to be handled by the client.

2.2 Solving a Puzzle

1. We receive a random number n. In this example we received $n = 66363$.

2. The code can be found in the following repository:
   `https://github.com/pfeifer-j/resilient_networks_ex4`

   We receive the following data:

```
Connected to net1-public.informatik.uni-hamburg.de:23001
Received: 687338
Sent: 700745
Received: Great. Here is your data:
   ***
  ** **
 **   **
 **   **          ****
 **   **        **   ****
 **  **        *   **   **
  **  *       *  **  ***  **
   **  *     *  **      **  *
    ** **  ** **         **
    **   **  **
   *            *
  *              *
 *    0     0    *
 *   /   @   \   *
 *   \__/ \__/   *
   *     W     *
     **     **
       *****
Connection closed
```

3. Guessing the first 10 bits correct takes on average $\frac{1}{0.5^{10}} = 1,024$ tries. We can calculate the expected computation duration using $\frac{1,024 Hashes}{863 Hashes/s} = 1.187s$.