Lab 10: ARP Spoofing and TLS

50.020 Security

Hand-out: April 18
Hand-in: 9 pm, April 29, 2019

1 Objective

By the end of this lab, you should be able to:

- Implement a simple HTTP API with basic authentication (similar to content of 50.012 Networks Lab 2. A copy of the Networks Lab 2 is uploaded in eDimension, just in case needed, particularly for those who didn't take 50.020 Networks)
- Use ARP spoofing to eavesdrop on third parties HTTP connections
- Create certificates for TLS/SSL connections
- Use your certificate to secure your HTTP connection

Note: The ARP spoofing part needs to be done in the lab

2 Implementing a simple HTTP-based messaging service

2.1 Basic system overview

- The goal is to implement a simple HTTP-based server that allows users to send and retreive messages
  - Use the previously used tutorial at `http://blog.luisrei.com/articles/flaskrest.html` if you need a flask refresher
- Messages are sent by a HTTP POST request to /message/<username>
  - It's enough to store submitted messages in a dict (non-persistent. Use SQLite or similar if you want persistence).
  - Example curl one-line: `curl -u guest:password -H "Content-Type: application/json"\ -X POST -d '{"message":"Hello World"}' http://localhost:5000/message/admin`
- Messages are retreived by a HTTP GET request to /messages (as JSON)
  - Example curl one-line: `curl -u guest:password http://localhost:5000/messages`
- All communication should use HTTP basic authentication with a username and password
  - You can hardcode both the username and password into the server

- We recommend to use flask to implement the server
- You should also implement a simple client (e.g. using Python and requests library) to POST one message to a username of your choice, and then retrieve messages
- Run both the server and client on your machine, and use wireshark to capture the traffic. What information can you obtain by eavesdropping? Note: you need to install wireshark if you haven't done before.

3 ARP Spoofing

- This part of the exercise requires you to be in the same local network as other users. We strongly recommend to use the wired network in the LEET lab. Make sure that your machine has a 10.0.1.x address, and that wireless connection is DISABLED.
- **DO NOT DO THIS PART ON SUTD_ STUDENT**. You will likely get your access locked for the next months.

3.1 ARP Spoofing with Ettercap

- Recall what we learned about ARP in 50.012 Networks (and/or check Wikipedia or similar).
- In this part, we will use *ARP spoofing* to redirect traffic between two victims via our machine.
- Start ettercap and familiarize yourself with the tutorial at `http://www.thegeekstuff.com/2012/05/ettercap-tutorial/`
- Your goal is to redirect an ongoing transmission from 10.0.1.20 (Target1) to 10.0.1.10 (Target2)
- Perform the attack as described to redirected the traffic
- What messages are sent out by your machine to perform the attack?
- Can you see the redirected messages (e.g. using wireshark)? Can you get the HTTP basic auth username and password?
  - If you can, feel free to leave a message on the server!
- Make sure to stop the attack again, as described in the tutorial
- Note: If multiple students are performing the attack at the same time, only the most recent attack will work!

4 TLS and HTTPS

- Follow the following guide to create a self-signed certificate `https://www.akadia.com/services/ssh_test_certificate.html` (or any reference you can find from Internet)
  - Note: The arguments in the tutorial are **very** outdated (key length, 3DES), so don't create real certificates like this

- We are now going to secure the server that you wrote earlier against the ettercap attack (hints: examples in the following link may be helpful: https://stackoverflow.com/questions/29458548/can-you-add-https-functionality-to-a-python-flask-web-server)
  - In particular: we wrap the HTTP socket connection using TLS.
- In your server, add the SSL context for the `app.run()`.
  - Use the self-signed certificate you just created
- In your HTTP client, change the server URL to reflect the use of HTTPS
  - You will most likely get an SSL error when connecting to your server. Why is that? See if it is possible to do *certificate pinning* in requests. Do not disable verification of certificates. (The implementation of *certificate pinning* is optional, will not be graded.)
- Try to capture the data transferred using Wireshark. Can you still see the passwords or messages?

5 Hand-in

- Submit your server and client code, and your server certificate and the private key
- Also submit a very short report file with
  - The details learned by eavesdropping on the connection between 10.0.1.20 and 10.0.1.10
  - Whether wireshark allowed you to see secret data for your server/client before/after using TLS