# Digital Forensics - Network Evidence Falsification

Jan Pawlus
*Brno University of Technology*

April 12, 2019

## 1 A digital crime frame-up

Idea for this project originated from network forensics puzzles[1], which were mentioned in both lectures and labs. In these puzzles, it is necessary to find information about commited digital crimes from captured packets (like leaking sensitive information from a company etc). Situations like these are not so simple in real life due to possibilities of using secure connection. But what if we actually do not use secure connection, leak sensitive information and frame somebody else for the crime as a bonus without any network trace?

This project demonstrates a possibility of doing such frame-up. The point is to send sensitive information from network to a remote server with spoofed *IP* and *MAC* address of different computer from the same network. The result is a set of captured packets where a forensic expert can see that sensitive information was leaked from different computer.

Unfortunately, I was unable to find any related resources for this topic - the most similar topics using *IP* and *MAC* spoofing focus mainly on attacks like *Man in the Middle*, which can be done mostly by *ARP cache poisoning* or *ARP spoofing* [1]. This approach would not work for framing somebody for a crime as *ARP spoofing* creates unquestionable evidence.

## 2 Solution proposal

First, it is needed to create an environment simulating a company network. For this purpose I chose two virtual machines (*Ubuntu*) connected to the internet via *bridged adapter* of the host computer (*Windows 10*). This setup allows me to operate with both virtual machines as two independent computers, whereas the host computer can capture all the packets heading for a *default gateway* from these virtual machines.

The network topology can be seen in the picture 1.

[1]Network forensics puzzles - `http://forensicscontest.com/puzzles`

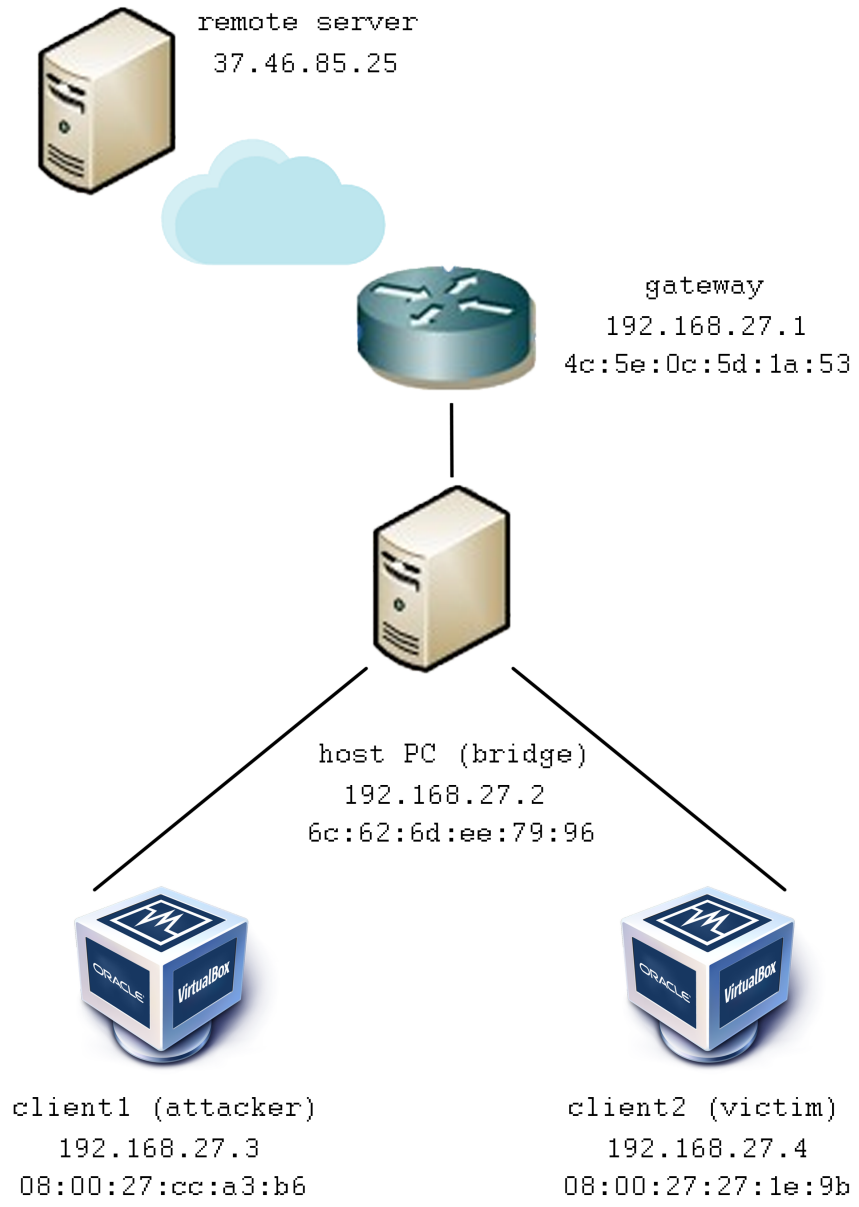


Figure 1: Network topology.

Afterwards, it is necessary to set up a remote server that will save the sensitive data. I have an access to a server of company I work for, which is sufficient for this purpose. There will be a program listening to a specific port and saving the incoming data.

The most important question is how the data will be transferred - whether to use *TCP* or *UDP* transporting protocol [2]. Taking into account the fact that the *IP* and *MAC* addresses are about to be spoofed, *UDP* fits to this purpose better - for using *TCP* it would be necessary to perform router's *ARP cache poisoning* so the attacker can receive *TCP* packets for connection establishment. Another advantage of *connectionless* protocol is that the remote server does not need to send any packets back, meaning that there is no evidence of even receiving the sensitive

data (it cannot be decided whether the remote server has anything to do with the crime or the data was sent to random *IP address*).

The transferred data will be encoded with *base64*.

# 3 Implementation

Because the server I have an access to operates on *Windows* and provides services with *.NET* technologies (and has no *Python* interpreter etc), the program was developed using `C#`. It is a very simple implementation that listens on specific port for *UDP* packets. When first chunk of data is received, a timeout is set so the end of connection is detected. Afterwards, packets' payloads are appended to each other, decoded from *base64* and saved as a file.

On the other side, a simple *Python* script was written with a help of *Scapy* library[2], which offers very easy way to create packets with spoofed *L2* and *L3* layers. The script loads specified file, encodes it with *base64*, creates safe-sized chunks and sends the packets.

The only problem remaining to deal with is to decide which *IP* and *MAC* addresses to spoof. First possibility is to use random values just to cover attacker's own steps but for the frame-up, it is necessary to obtain another PC's *MAC* and *IP* addresses. An immediate solution is to use *ARP* spoofing - pretending to be the gateway and asking other computers which *MAC* address they use. Another possibility is to use *ICMP* protocol. Nevertheless, both of these approaches create evidence.

The best way to obtain another computer's *IP* and *MAC* address is to wait until a victim sends a broadcast *ARP* packet, asking for (usually the gateway's) *MAC address*. Because this packet is broadcasted, the attacker can sniff it and use its values for the attacking script. This packet is broadcasted usually when the victim computed is just powered on or when the victim's *ARP* cache times out, as picture 2 shows.

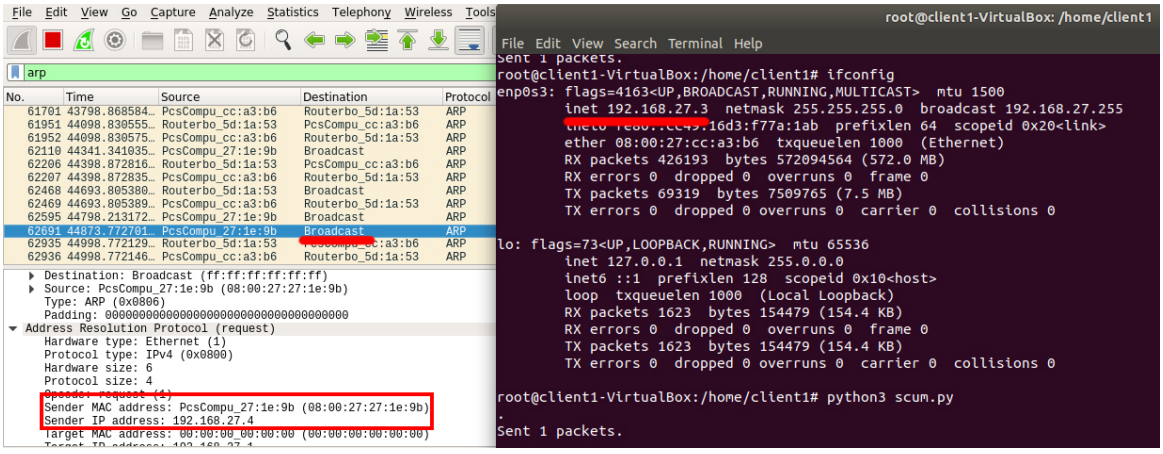


Figure 2: Sniffing for victim's broadcasted *IP* and *MAC* addresses.

[2] *Scapy* - `https://scapy.net/`

# 4 Evaluation

After executing the attacking script while capturing packets on the host PC, an investigator can see that the suspicious packets seem to be sent from the victim's PC instead of the attacker's PC (picture 3).

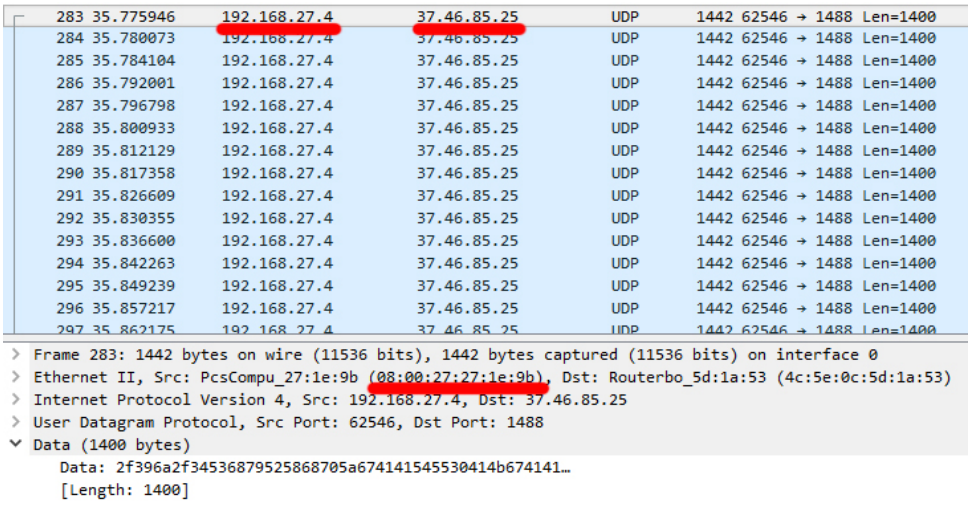


Figure 3: Suspicious *UDP* stream sent from attacker's PC

Moreover, because the connection is not secure, when following the *UDP stream*, the *base64*-encoded file can be seen and easily reconstructed (in this case, file `sensitiveData.jpg` was sent). In the meantime, packets are reconstructed and saved into a file on the remote server (picture 4).

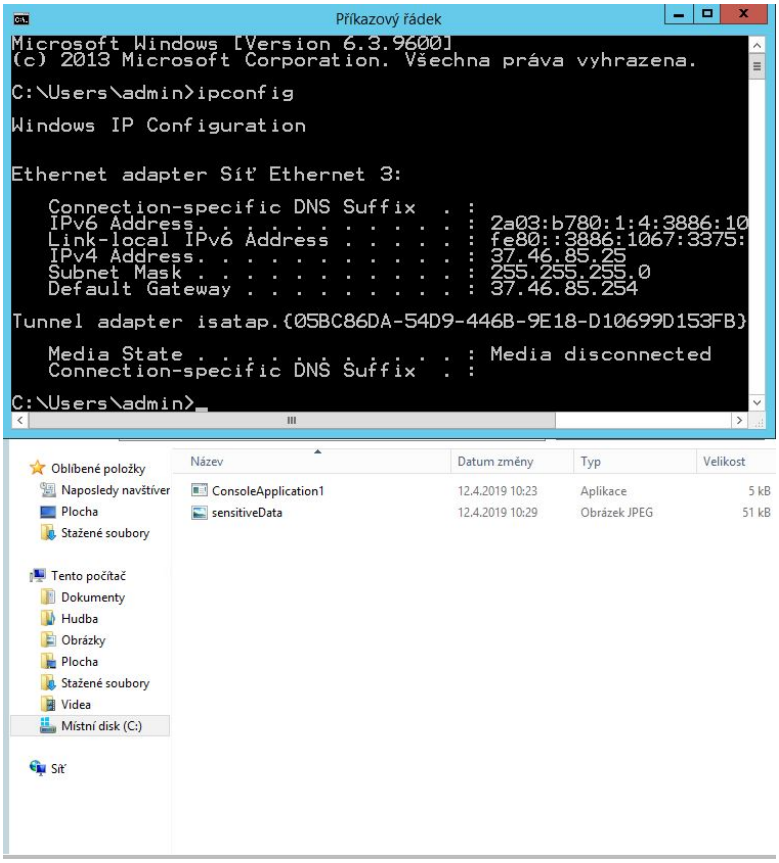


Figure 4: Remote server' view

# 5 Conclusion

This project demonstrated how sensitive data can be sent from attacker's PC while framing somebody else (evidence falsification). When investigating captured packets, there is no evidence connecting the attacker to the crime as the attacker does not need to send a single packet. This is recorded in attached `.pcap` file. The attacking script can be seen in `attack.py`, the remote server's program in `server.cs`.

# References

[1] Venkatesh Sundar. *How to Prevent ARP Spoofing Attacks?* `https://www.indusface.com/blog/protect-arp-poisoning/`.

[2] Steve's Internet Guide. *TCP vs UDP - What's The Difference?* `http://www.steves-internet-guide.com/tcp-vs-udp/`.