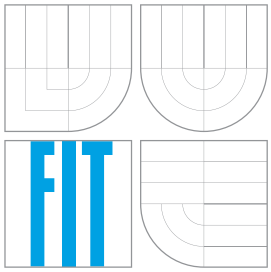
BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

AUTOMATION OF MITM ATTACK ON WIFI NETWORKS

AUTOMATIZACE ÚTOKU MITM NA WIFI SÍTÍCH

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR
AUTOR PRÁCE
MARTIN VONDRÁČEK

SUPERVISOR
VEDOUCÍ PRÁCE
Ing. JAN PLUSKAL

BRNO 2016

Bachelor's Thesis Specification/18596/2015/xvondr20

Brno University of Technology - Faculty of Information Technology

Department of Information Systems Academic year 2015/2016

Bachelor's Thesis Specification

For: **Vondráček Martin**
Branch of study: Information Technology
Title: **Automation of MitM Attack on WiFi Networks**
Category: Networking

Instructions for project work:

1. Study different kinds of security approaches used in wireless networks. Focus on known vulnerabilities of individual methods and on tools for exploiting these vulnerabilities.
2. Consider the possibility of impersonification of specified AP in case that attack on given device is not available.
3. Utilize existing tools from items 1 and 2, and choose the most appropriate one or a set of them, which will be able to realize automatic attack on chosen network.
4. Implement a tool capable of such automation. This tool will be able to choose the most suitable attack or a sequence of them.
5. Test the solution during experiments in laboratories. Evaluate the success rate of implemented solution against different kinds of existing AP.

Basic references:

- Callegati, F., Cerroni, W. & Ramilli, M., Man-in-the-middle attack to the HTTPS protocol. IEEE Security and Privacy, 7(1), p.78-81. 2009.
- Dierks, T. & Rescorla, E., 2008. RFC 5246 - The transport layer security (TLS) protocol - Version 1.2. In *Network Working Group, IETF*. pp. 1-105.

Requirements for the first semester:
Items 1, 2 a 3.

Detailed formal specifications can be found at http://www.fit.vutbr.cz/info/szz/

The Bachelor's Thesis must define its purpose, describe a current state of the art, introduce the theoretical and technical background relevant to the problems solved, and specify what parts have been used from earlier projects or have been taken over from other sources.

Each student will hand-in printed as well as electronic versions of the technical report, an electronic version of the complete program documentation, program source files, and a functional hardware prototype sample if desired. The information in electronic form will be stored on a standard non-rewritable medium (CD-R, DVD-R, etc.) in formats common at the FIT. In order to allow regular handling, the medium will be securely attached to the printed report.

Supervisor: **Pluskal Jan, Ing.**, DIFS FIT BUT
Beginning of work: November 1, 2015
Date of delivery: May 18, 2016

L.S.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

Dušan Kolář
Associate Professor and Head of Department

Abstract

This bachelor's thesis aims at research concerning a security of wireless networks. It delivers a study of widely used network technologies and principles of wireless security. Analysed technologies and security algorithms suffer weaknesses that can be exploited to perform the MitM attack. The thesis includes an overview of available tools focused on exploiting these individual weaknesses.

The outcome of the thesis is the *wifimitm* package and the *wifimitmcli* CLI tool, both implemented in Python. The package provides a functionality for automated MitM attack and can be used by other software. The *wifimitmcli* tool is capable of performing a successful fully automated attack without any intervention from an attacking person. This research can be used for automated penetration testing and for forensic investigation.

Abstrakt

Tato bakalářská práce se zaměřuje na výzkum v oblasti bezpečnosti bezdrátových sítí. Práce přináší studii široce využívaných síťových technologií a principů zajištění bezpečnosti bezdrátových sítí. Analyzované technologie a způsoby zabezpečení trpí slabinami, které mohou být zneužity k provedení útoku MitM. Práce zahrnuje přehled dostupných nástrojů zaměřených na využití jednotlivých slabin.

Výsledkem této práce je balíček *wifimitm* a CLI nástroj, oba implementované v jazyce Python. Balíček poskytuje funkcionalitu pro automatizovaný útok MitM a může být použit jako součást dalšího software. Nástroj *wifimitmcli* je schopen úspěšného provedení plně automatizovaného útoku bez jakéhokoli zásahu útočící osoby. Tento výzkum nachází využití v oblasti automatizovaných penetračních testů a forenzního vyšetřování.

Keywords

Wi-Fi Machine-in-the-Middle, wifimitm, wifimitmcli, MitM, WLAN, Wi-Fi, DHCP, ARP, Neighbor Discovery, DNS, HTTP, HTTPS, WEP, WPA, WPA2, WPS, DHCP Spoofng, DHCP Snooping, ARP Spoofng, Dynamic ARP Inspection, IPv6 Neighbor Spoofng, Neighbor Discovery Inspection, DNS Spoofng, wifiphisher, upc_keys, Aircrack-ng suite, airmon-ng, airodump-ng, aireplay-ng, aircrack-ng, Scapy, dsniff, arpspoof, Yersinia, Framework for Man-In-The-Middle attacks, MITMf, dnsspoof, Python, Bash

Klíčová slova

Wi-Fi Machine-in-the-Middle, wifimitm, wifimitmcli, MitM, WLAN, Wi-Fi, DHCP, ARP, Neighbor Discovery, DNS, HTTP, HTTPS, WEP, WPA, WPA2, WPS, DHCP Spoofng, DHCP Snooping, ARP Spoofng, Dynamic ARP Inspection, IPv6 Neighbor Spoofng, Neighbor Discovery Inspection, DNS Spoofng, wifiphisher, upc_keys, Aircrack-ng suite, airmon-ng, airodump-ng, aireplay-ng, aircrack-ng, Scapy, dsniff, arpspoof, Yersinia, Framework for Man-In-The-Middle attacks, MITMf, dnsspoof, Python, Bash

Reference

VONDRÁČEK, Martin. *Automation of MitM Attack on WiFi Networks.* Brno, 2016. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Pluskal Jan, Briffa Johann.

Rozšířený abstrakt

Tato bakalářská práce se zaměřuje na výzkum v oblasti bezpečnosti bezdrátových sítí. Cílem výzkumu byl návrh, implementace a testování nástroje, který by byl schopen provedení plně automatizovaného útoku Man-in-the-Middle na bezdrátových sítích bez jakéhokoli zásahu útočící osoby. Úspěšné provedení útoku MitM umožňuje útočníkovi nejen odposlouchávat, ale také upravovat probíhající komunikaci oběti. Práce přináší studii široce využívaných síťových technologií a principů zajištění bezpečnosti bezdrátových sítí. Analyzované technologie a způsoby zabezpečení trpí slabinami, které mohou být zneužity k provedení útoku MitM.

Práce analyzuje metody a nástroje zaměřené na penetrační testování, možnosti zachycení a podvržení síťové komunikace, prolamování hesel a hacking obecně. V současné době jsou dostupné specializované nástroje zaměřené na jednotlivé slabiny bezpečnostních principů aplikovaných v bezdrátových sítích. Obdobně jsou k dispozici nástroje zaměřené na jednotlivé kroky nutné k úspěšnému provedení útoku MitM. V situaci, kdy je vybraná bezdrátová síť dostatečně zabezpečena po technické stránce, je možné se zaměřit na samotné uživatele sítě, kteří mohou být slabým článkem vedoucím k prolomení zabezpečení sítě. Z tohodo důvodu práce dále analyzuje možnosti impersonifikace a phishingových útoků za účelem získání neautorizovaného přístupu do sítě.

Strategie útoku byla vyvinuta s využitím získaných znalostí, citovaných výzkumů a praktických zkušeností z provedených experimentů. Strategie je schopna volby jednotlivých kroků, které využívají dostupné nástroje a vedou k úspěšnému útoku MitM.

Výsledkem této práce je balíček *wifimitm* a CLI nástroj, oba implementované v jazyce Python. Balíček poskytuje funkcionalitu pro automatizovaný útok MitM a může být použit jako součást dalšího software. Nástroj *wifimitmcli*, využívající funkcionalitu implmentovaného balíčku, zajišťuje jednotlivé kroky útoku. Výsledná distribuce software obsahuje řadu náležitostí pro zajištění pohodlného použití, například instalační skripty, kontrolu požadavků na software, instalaci Python balíčku a nástroje *wifimitmcli*, soubor *README.md* a manuálovou stránku.

Automatizovaný nástroj *wifimitmcli*, tedy i balíček *wifimitm*, byl testován v průběhu experimentů s využitím dostupného vybavení. Na základě výsledků experimentů se ukázalo, že implementovaný software je schopen úspěšného provedení plně automatizovaného útoku bez jakéhokoli zásahu útočící osoby.

Software, který je výsledkem tohoto výzkumu, může být použit pro potřeby forenzního vyšetřování. Nástroj schopný automatizovaných útoků MitM na bezdrátových sítích může být dále využit ke zdokonalení zabezpečení sítí díky automatické detekci jejich slabin. Z tohoto pohledu může být *wifimitmcli* považován za automatizovaný penetrační nástroj.

Automation of MitM Attack on WiFi Networks

Declaration

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the supervision of Ing. Jan Pluskal from the Brno University of Technology, Faculty of Information Technology and Dr Johann A. Briffa from the University of Malta, Faculty of Information and Communication Technology. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Martin Vondráček
May 19, 2016

Acknowledgements

I would like to thank Ing. Jan Pluskal and Dr Johann A. Briffa for the opportunity to work on this research under their supervision. I deeply appreciate their help and professional advices. I am very grateful to Dr Johann A. Briffa for accepting to be my supervisor which allowed me to study at the University of Malta. I would also like to thank Ing. Jan Pluskal for encouraging my interest in information security.

Contents

Chapter 1

Introduction

This bachelor's thesis is focused on security of wireless networks. The aim of this research is to design, implement and test a tool able to automate the whole process of Man-in-the-Middle attack on Wireless Local Area Networks. This way, using the automated tool would not require any action from the user of the tool.

The Man-in-the-Middle is definitely a very dangerous network attack. A successful realization of this attack allows not only eavesdropping on all the victim's network communication but also spoofing the communication.

Concerning the Wireless Local Area Networks, one of their main advantages is the possibility of user's convenient mobility within the range of the network. However, this could be considered their weakness at the same time. Wireless networks transfer information between communicating devices using a shared medium – electromagnetic signal. Based on this concept, the network could be attacked by devices which are within the signal range. The attacker therefore does not need a direct access to the network devices.

Because the MitM attack exploits weaknesses in currently used network technologies, it is essential to focus on significant network technologies in chapter 2. Outlined technologies are widespread, but unfortunately, in current form, they do not provide sufficient security solutions.

For the chosen implementation of the Man-in-the-Middle attack, it is important for an attacker to be in the same WLAN as a victim. For this reason, chapter 3 of this research focuses on principles of wireless networks, security possibilities and especially on exploitable weaknesses of individual security algorithms.

Upon successful connection to the network, the attacker needs to adjust a network topology. For this purpose, weaknesses of several network technologies can be exploited. Possible ways of tampering the network topology are described together with available countermeasures against specific attacks in chapter 4. From the point when the attacker is connected to the network and topology is successfully modified, the attacker can start capturing or modifying all the victim's network traffic.

A detailed characteristic of the Man-in-the-Middle attack is in chapter 5. This chapter further describes that even a secure communication using HTTPS can be successfully attacked, captured and modified.

Finally, the software product based on the whole research is described in chapter 6. Design and implementation work on this thesis resulted in a Python package and a CLI tool providing functionality for the automated attacks. The implemented package was named *wifimitm* and the CLI tool was named *wifimitmcli*. Emphasis was placed on possibilities of further incorporation of the developed tool, convenient modification and distribution.

Chapter 2

Significant network technologies

This chapter provides a basic description of the network technologies which are currently used in wireless networks and are important for the remainder of this thesis, namely, Dynamic Host Configuration Protocol in the section 2.1, Address Resolution Protocol in the section 2.2 and Neighbor Discovery in the section 2.3.

Following technologies, which find a significant utilization, unfortunately suffer from security weaknesses in their protocols. These security flaws can be used in the process of Man-in-the-Middle attack. Individual attacks, exploiting the flaws of these protocols, with possible countermeasures are outlined in sections 4.1 to 4.3.

2.1 DHCP

Dynamic Host Configuration Protocol is used to provide the network device with a suitable configuration without need for intervention from the user. Because of the use of DHCP in WLAN networks, users can comfortably connect to network and in a very short time start to communicate.

The functionality of DHCP is based on the client-server model. The DHCP server manages the assignment of network configuration details to individual clients. Computers connecting to the network take the role of clients. The client needs to run DHCP client software for the purpose of DHCP communication. After successful connection to network, the client inquires DHCP server for configuration details. For the correct network connectivity of the client, the following configuration details are obtained from the DHCP server:

- client's IP address from an appropriate range,
- subnet mask,
- IP address of the default gateway,
- IP address of DNS server for domain name resolution.

DHCP server's reply can contain additional configuration details. Sequence of messages exchanged between DHCP client and servers is in appendix A (Droms 1997). More information can be found in the literature (Droms 1997).

2.2 ARP

ARP refers to the Address Resolution Protocol, which provides mapping of IP address to the corresponding MAC address of the device in local area network based on IPv4. In networks based on IPv6, mapping of IP and MAC addresses is provided by Neighbor Discovery, which is described in section 2.3.

Communication in the context of ARP is composed of ARP Request and ARP Reply messages. Devices on the LAN need to be aware of individual MAC addresses for connectivity. In the case that some device needs to send data to a given IP address, the device needs to obtain the corresponding destination's MAC address. If the sender does not have information about the mapping of IP and MAC addresses, it transmits an ARP Request to the LAN. This message is sent to all devices on the local network, requesting the device specified by IP address to reply with its MAC address. The appropriate device replies using an ARP Reply message and announces the mapping of its IP and MAC address (Plummer 1982).

ARP Request broadcast on the local network can be used by other devices to record IP and MAC address mappings. Network devices store information gathered from ARP communication in an ARP table, also referred to as the ARP Cache. If the sender has information concerning required mapping in its ARP table, the sender is able to immediately send data to destination (Plummer 1982).

2.3 Neighbor Discovery

IPv6 utilizes features of Internet Control Message Protocol version 6 (ICMPv6) in order to map devices in the local network with Neighbor Discovery (ND). To be able to communicate in IPv6 network, a device needs information about IP and MAC address mapping in the same way as in IPv4.

IPv6 ND provides similar functionality as ARP in IPv4 networks. In terms of this work, especially important is the Neighbor Solicitation (NS), and Router Advertisement (RA). Messages of NS are transmitted in network when device is inquiring the MAC address of a device specified by IPv6 address. After delivery of NS, the appropriate device announces its MAC address with Neighbor Advertisement (NA). Messages of NS can be also used to determine reachability of specified point in network. Messages of RA are periodically sent by the router connected to a local network. These messages are used to continuously announce the existence of router and other details (Cisco Systems, Inc. 2012, p. 81).

2.4 Summary

In the chapter Significant network technologies, three current network technologies relevant for the MitM attack were introduced. DHCP provides automated assignment of network configuration. ARP is used to obtain the mapping information about IP and MAC addresses of the devices in IPv4 networks. Neighbor Discovery is used in IPv6 networks for similar purpose as ARP.

Chapter 3

Wireless Local Area Networks

In section 3.1 of this chapter, the functionality of Wireless Local Area Networks (WLANs) is outlined as an introduction to the principles of wireless communication. Due to the fact that the topic of this thesis is strongly related to the security of WLANs, principles of providing security in WLANs are analyzed in sections 3.2 to 3.5. Considering following analysis of security principles, it is important to define network security in the first place. Security of network communication can be divided into following areas (Halsall 2005, p. 633):

Integrity
Securing that network communication was not spoofed.

Privacy
Ensuring that transferred information is not available to any unauthorized party.

Authentication
Communication is realized with authenticated party.

3.1 Concept of WLAN

The terms WLAN and Wi-Fi are currently used for wireless networks, where particular devices communicate through electromagnetic signal and their communication meets the standard IEEE 802.11 (Halsall 2005). However, the label Wi-Fi® was introduced for certification of devices that are compatible for communication in wireless networks. The certified devices can be marked *Wi-Fi CERTIFIED™*, which is a trademark of the Wi-Fi Alliance®, originally Wireless Ethernet Compatibility Alliance (Wi-fi.org 2016).

Radio communication of electronic devices can be characterized by the frequency. A frequency range is typically licensed, that is, permission is necessary for using a device communicating on the licensed frequency (CTU.cz 2016).

Wireless networks use a special unlicensed bandwidth for their radio communication. There are special frequency ranges in which the devices can communicate when their owner does not possess a special license for usage of the given frequency. These special intervals comprise of frequencies around 2,4 GHz and 5 GHz. There are other specific requirements for operation of radio devices on the unlicensed frequency, for example, a maximal output power (CTU.cz 2016).

Due to the fact that there is a frequency range designated to WLAN networks, interference can occur in areas with a high density of WLAN networks. However, it is not possible to restrict operations in these networks because a non-restrictive attitude is essential for

the unlicensed band (CTU.cz 2016). On the basis of IEEE 802.11 standard (IEEE-SA 2012) a wireless network can operate in one of two modes:

- infrastructure
- ad-hoc

The infrastructure mode introduces one main radio device called the Access Point (AP). The network then contains several devices communicating with the AP, called Stations (STAs). In wireless networks connected to the Internet, the AP ensures a radio communication with devices in local area network as well as a communication with the Internet. In these networks, the communication takes place only between an STA and the AP, never between the STAs themselves (Halsall 2005). Wireless networks operating in ad-hoc mode do not use an AP. Networks in ad-hoc mode include STAs which communicate with each other directly.

3.2 WEP

Wired Equivalent Privacy is a wireless network security algorithm introduced in 1997 as a part of the IEEE 802.11 standard (Halsall 2005, p. 665), (IEEE-SA 2012, pp. 1167–1169). At this point, WEP is deprecated and superseded by subsequent algorithms. WEP suffers from weaknesses and, therefore, it has been broken (Fluhrer, Mantin, and Shamir 2001). There are already implemented tools to provide access to wireless networks secured by WEP available (Tews, Weinmann, and Pyshkin 2007).

Based on the IEEE 802.11 standard (IEEE-SA 2012), STAs need to perform a successful authentication and association to join the wireless network. Regarding WEP secured WLAN, authentication can be either Open System Authentication (OSA) or Shared Key Authentication (SKA) (IEEE-SA 2012, pp. 1170–1174). In the case of WEP OSA, any STA is able to successfuly authenticate to the AP. Sequence of Open System Authentication is in fig. 3.1. Detailed information concerning individual frames can be found in the literature (Robyns 2014, pp. 4–10), (IEEE-SA 2012, pp. 404–437).

WEP SKA provides authentication and security of transferred communication using a shared key. The AP and all STAs in the network use the same shared key for encryption and decryption of communication. Sequence of WEP Shared Key Authentication is shown in fig. 3.2. Authentication of a STA, which is interested in connecting to the network, is done via three-way handshake with the following principle (Halsall 2005, p. 665):

1. STA requests a connection to the WEP SKA secured wireless network, sends message to AP and identifies itself.
2. AP replies with a plaintext challenge text to STA.
3. STA encrypts received challenge text using shared key previously set for this network, encrypted text is sent back to AP.
4. AP decrypts the encrypted challenge text received from STA, if decrypted text and original challenge text are the same, STA possesses the correct key for network and it is successfully authenticated and connected.

Confidentiality of transferred data is ensured by encryption using the Rivest Cipher 4 (RC4) stream cipher. RC4 uses concatenation of shared key and Initialization Vector (IV)

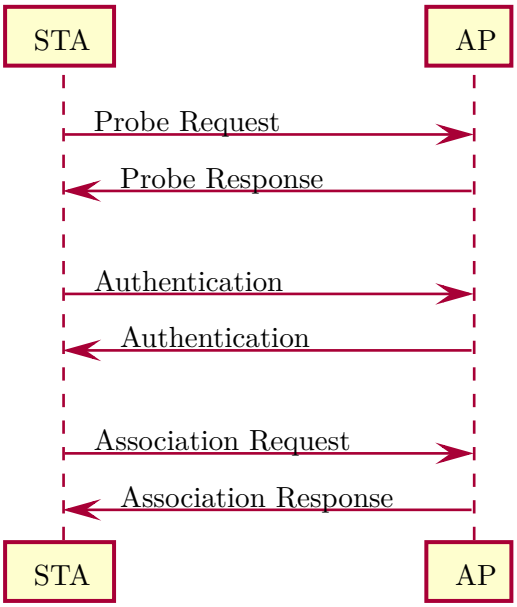


Figure 3.1: Sequence of WEP Open System Authentication

for generation of a sequence of values – a stream. The generated stream is used for encryption and decryption of transferred data. Encrypted data is sent over the network with enclosed initialization vector used for its encryption. After delivery of encrypted data, IV can be used together with the shared key to successfully decrypt the data. For more information concerning this topic see (Halsall 2005, pp. 665–667). The integrity of data is provided by the Cyclic Redundancy Check (CRC). Methods used for cracking access to WEP secured networks are based on analysis of transferred data with corresponding IVs.

3.3 WPA

Wi-Fi Protected Access® was developed in 2003 by the Wi-Fi Alliance as a reaction to increasing number of security flaws in WEP. Introduced WPA was a subset of IEEE 802.11i standard (Liu, Jin, and Wang 2010, p. 1), (Jelínek 2010, p. 60), (Robyns 2014, p. 22). Change from WEP to WPA security algorithm was possible for some client devices and APs using a firmware update.

Main security improvement of WPA is the Temporal Key Integrity Protocol (TKIP). The principle of TKIP is an algorithm where every packet uses its own generated secret 128-bit key. WPA also comes with advanced methods to provide data integrity. From the security configuration point of view, WPA is differentiated between personal and enterprise use. These two WPA forms are referred to as WPA Personal and WPA Enterprise. The difference is distinguishable mainly in the ways of authentication, where personal mode relies on a Pre-Shared Key (PSK) key and enterprise mode uses an authentication server.

Main flaws of WPA security algorithm can be identified in the case of using PSK for generation of other keys. In the beginning of client device's communication, unsecured exchange of confidential information is performed during four-way handshake. An attacker can obtain this unsecured communication and use it for consecutive cracking of the PSK. Concerning the effort of gaining unauthorized access to the secured network, an attacker can

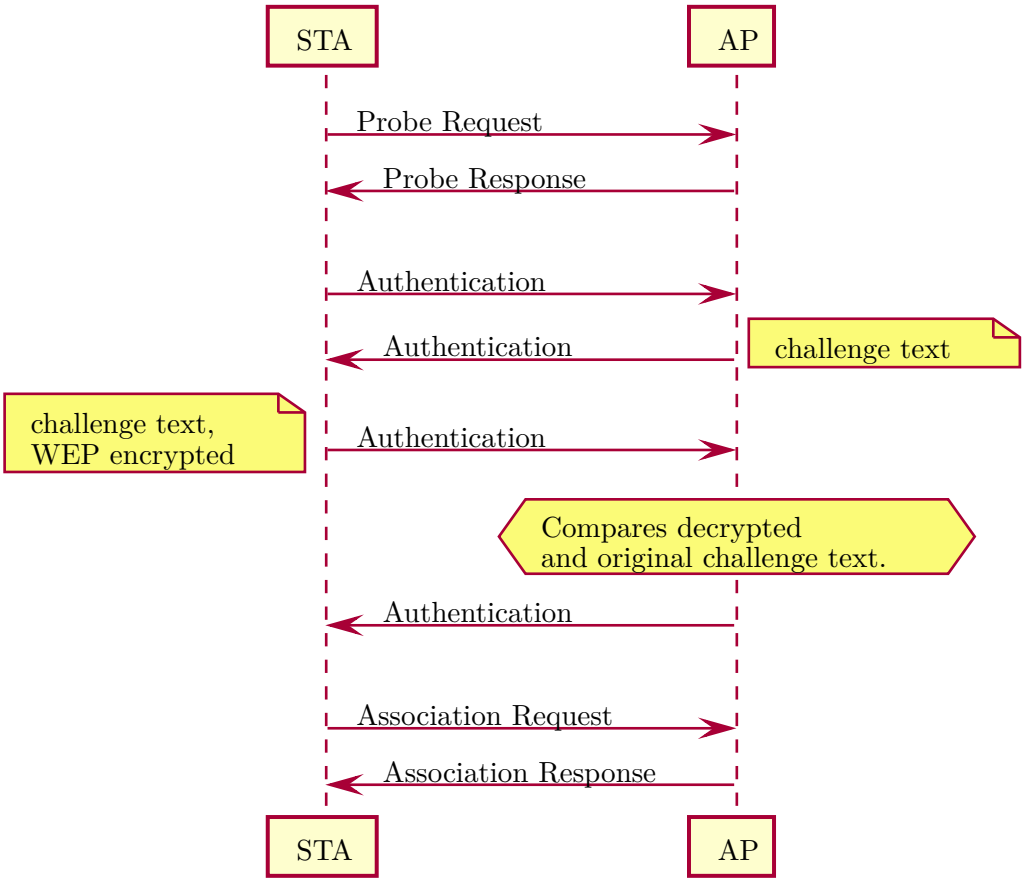


Figure 3.2: Sequence of WEP Shared Key Authentication

focus on dictionary attacks. The speed of dictionary attacks can be increased for example by precomputing the Pairwise Master Keys (PMKs). Precomputation of PMKs can be performed offline and before the individual network attack. This approach can be described as space-time tradeoff. With multithreaded systems and GPUs, precomputation of PMKs is a significant improvement of this dictionary attack. More details about possibilities of precomputation of PMKs can be found in the literature (Liu, Jin, and Wang 2010, p. 3). A detailed description of WPA is available in the literature (Jelínek 2010).

3.4 WPA2

Security algorithm Wi-Fi Protected Access II is a successor of WPA. WPA2™ was introduced in 2004 and provides higher level of security in comparison to previous security algorithms. Security principles utilized in WPA2 are defined in an amendment to IEEE 802.11 standard named IEEE 802.11i (IEEE-SA 2004), (Jelínek 2010, p. 60), (Robyns 2014, p. 22).

Due to the higher complexity of securing mechanisms used in WPA2, this algorithm was not suitable for some devices in the time of its introduction. Similarly as WPA, WPA2 distinguishes between personal and enterprise modes. The main advantage of WPA2 is a utilization of Advanced Encryption Standard using symmetric block cipher.

Weaknesses of the Pre-Shared Key algorithm remain significant also for the WPA2.

Information exposed during the four-way handskahe can be used for the dictionary attack, which can also be improved by PMKs precomputation (Kumkar et al. 2012, pp. 37–38), (Liu, Jin, and Wang 2010, p. 3).

3.5 WPS

A very important security flaw in wireless networks secured by WPA or WPA2 is functionality called Wi-Fi Protected Setup™. This technology was introduced in 2006 with aim to provide an easy and secure way of connecting to the network.

For connection to the correctly secured network, it is possible to use a special Personal Identification Number (PIN). The process of connecting to the properly secured network by providing PIN is however very prone to brute-force attacks (Viehböck 2011a), (Viehböck 2011b), (Heffner 2011). Due to the fact that WPS is a common feature in today's access points and that WPS is usually turned on by default, WPS can be a very common security flaw even in networks secured by WPA2 with a strong password. Currently, there are already available automated tools for exploiting WPS weaknesses, e.g., *Reaver Open Source*[1].

3.6 Specific security flaws

Apart from the main and general security flaws applied to the majority of wireless networks, many networks are still vulnerable to basic security negligence. These flaws are commonly caused by a user who is not aware of the security aspects and who has not set up configuration correctly.

Newly purchased access points usually use WPA2 security by default. Currently, many access points can be found using default passwords not only for wireless network access, but even for AP web administration. There are already implemented tools, which exploit relations between SSIDs and default network passwords, e.g., *upc_keys*[2] by Peter Geissler. These tools could be used in an attack on the network with default SSID to improve dictionary attack using possible passwords.

3.7 Tools for accessing wireless networks

Security algorithms described in previous sections suffer from weaknesses which can be exploited to access wireless networks. Already implemented tools focused on these vulnerabilities are outlined in this section. Described *Aircrack-ng suite* focuses on cracking of networks, while *wifiphisher* utilizes phishing attacks and social engineering.

Aircrack-ng suite

In order to access secured wireless networks, *Aircrack-ng suite* is considered a reliable software solution. *Aircrack-ng suite* is a collection of individual tools, which are aimed at assessing WLAN security.

The *airodump-ng* tool is capable of packet capture of raw wireless communication frames. It is also able to provide a summary of detected APs and STAs in a format convenient for

[1] URL: https://code.google.com/archive/p/reaver-wps/
[2] URL: https://haxx.in/upc-wifi/

further processing. Concerning attacks to the WEP secured networks, *airodump-ng* is able to collect IVs.

Airmon-ng is a script focused on the process of managing monitor mode of a wireless interface. It is also able to detect and stop running processes that could interfere with the other tools from the *Aircrack-ng suite*.

Injection of specific wireless frames can be done using the *aireplay-ng* tool. *Aireplay-ng* offers several injection attacks, e.g., Deauthentication, Fake authentication, ARP Request replay attack, KoreK chopchop attack, Fragmentation attack and Cafe-latte attack. It can be also used for injection test to determine, whether the attacker's device is capable of injecting packets and to analyze conectivity to the APs.

Another constituent part of the *Aircrack-ng suite* is a tool called *aircrack-ng*, which is focused on WEP, WPA and WPA2 PSK key cracking. *Aircrack-ng* can crack WEP key from a packet capture provided by *airodump-ng*. Cracking of WEP key depends on the capture of IVs. Cracking of WPA and WPA2 PSK is done by a dictionary attack and depends on the capture of four-way handshake.

Successful attacks on WEP and WPA/WPA2 utilizing *Aircrack-ng suite* can be found for example in (Kumkar et al. 2012, pp. 34–38) and in (Jelínek 2010, pp. 73–86). More information about the entire *Aircrack-ng suite* including tutorials for individual attacks can be found on its website (d'Otreppe 2016).

wifiphisher

In the case that an attacked network is secured properly, other attacking methods can be used in order to access the network. Even if the network uses WPA2 with a very strong and regularly changed password, the weakest part of the security can be legitimate users of the network. From this point of view, phishing attacks and social engineering attacks have a significant role. Success of these attacks strongly depends on security awareness of users.

The *wifiphisher* tool focuses on automated phishing attacks by impersonating attacked Access Point. Aim of this tool is to obtain a network password from users of the network. *Wifiphisher* is able to continuously deauthenticate STAs from selected AP. At this point, victims are not able to stay correctly connected to the genuine AP. At the same time, *wifiphisher* creates an AP which impersonates the attacked one. Depending on the individual setup of victim's device and power of signal of attacker's AP, victim's STA could even connect to the attacker's AP automatically. When the victim is disconnected from the genuine network, victim could try to select network manually and connect to the attacker's network. After the victim connects to the attacker's network, the victim receives selected phishing website.

Wifiphisher provides phishing website templates, e.g., browser connection reset requiring network password. *Wifiphisher* is implemented in Python language and more information about this tool can be found on its repository website[3].

3.8 Summary

The chapter Wireless Local Area Networks introduced main aspects of WLANs. Wireless networks can operate in an infrastructure mode with an AP and STAs, or in an ad-hoc mode consisting only of STAs.

[3]URL: https://github.com/sophron/wifiphisher

As described in section 3.2, WEP is a security algorithm which has severe security flaws. Concerning current state of technology, WLAN utilizing WEP should not be considered a sufficiently secured network. In the case of using a WEP secured network, users need to be aware of the fact that their communication can be easily eavesdropped and spoofed.

WPA and WPA2 provide significantly better level of network security. However, an attacker can easily obtain a four-way handshake, which is required for an offline dictionary attack. If the network uses WPA/WPA2 PSK, it is important to remember that the network is only as secure as strong is its key.

Considering the WPS, there are not many cases when its benefits would compensate the possible risks. Even if the network uses a very strong PSK, enabled WPS could be the weakest part of the chain, letting attackers in the network.

In section 3.7, the *Aircrack-ng suite* was introduced together with its important tools, e.g., *airodump-ng*, *airmon-ng*, *aireplay-ng* and *aircrack-ng*. This collection of tools can be effectively used by an attacker not only for a manual attack, but also as a part of other advanced tools. The *wifiphisher* tool can be used for a phishing attack to obtain the network password.

Chapter 4

Tampering network topology

The following text is focused on analysis of current state and possibilities of intentional unauthorized modification of network topology. Tampering could be performed at the moment when an attacker is successfully connected to the targeted wireless network.

4.1 DHCP Spoofing

DHCP Spoofing generates fake communication which is used for providing dynamic network configuration for devices in current network. Principles of DHCP are described in section 2.1. This attack can also be reffered to as Rogue DHCP (Duangphasuk, Kungpisdan, and Hankla 2011, p. 288).

An attacker can perform this kind of attack in order to provide devices in the network with special and intentional configuration, most often a fake default gateway address or DNS address. If the attacker manages to change the victim's default gateway to the attacker's address, the victim will route all its traffic through the attacker's device. At this point, the attacker is in a network topology suitable for the Man-in-the-Middle attack. If the attacker manages to impersonate the victim's DNS, he is then able to take control of all victim's domain name resolutions (Callegati, Cerroni, and Ramilli 2009, p. 79).

During this attack, the attacker runs his own DHCP server. At this point, the local network contains two DHCP servers, which are willing to offer configuration. Schema of this network attack can be seen in fig. 4.1. Sequence of this attack is shown in fig. 4.2. DHCP Spoofing progresses based on the following principle:

1. An attacker listens for DHCP communication and awaits a request for assigning the configuration.
2. DHCP client at client's device sends a request for configuration details to the local network. In fig. 4.2, the Client broadcasts DHCP Discover message.
3. Attacker's fake DHCP server creates a message and tries to reply to the client faster than the genuine DHCP server. This attack assumes that the attacker's DHCP server is able to reply faster than the real DHCP server. Speed difference can be improved by making the real DHCP server busy and overloaded.
4. In the case of successful attack, the client receives spoofed network configuration. Client accepts this configuration and the address of default gateway and optionally address of DNS is changed.

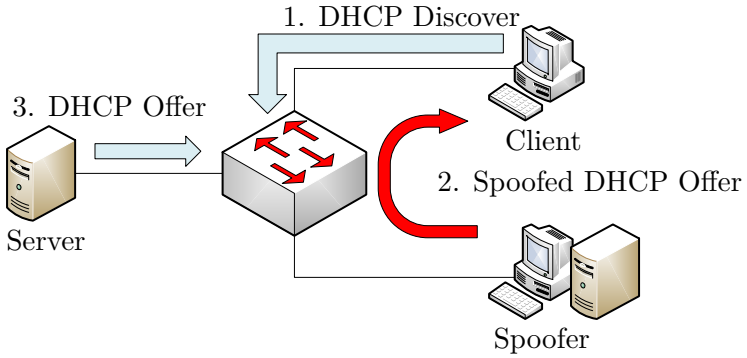


Figure 4.1: Schema of DHCP Spoofing

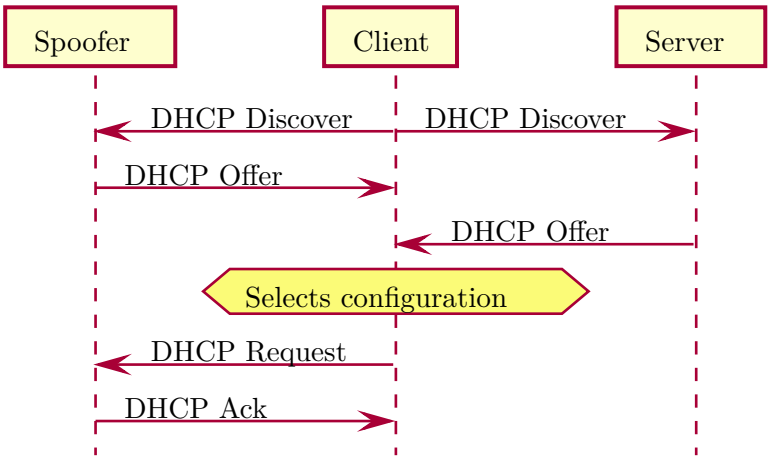


Figure 4.2: Sequence of DHCP Spoofing (rest of the communication is omitted)

DHCP Snooping

A possible countermeasure against DHCP Spoofing attack is a method called DHCP Snooping. By applying this method, network devices are configured in a way, that they are able to detect spoofed DHCP message.

Important network devices accept DHCP messages coming from connections linked to the genuine DHCP server. DHCP messages coming from connections which are not linked to the genuine DHCP server are declined. This means that individual network connections on individual devices are divided into groups and marked as trusted or untrusted (Cisco Systems, Inc. 2016, pp. 54-2). If the network contains an unknown DHCP server on an untrusted connection, the server is reffered to as *spurious DHCP server*. The detection of spoofed DHCP messages can result in, for example, disconnection of the attacker and notifying the network administrator. Schema of defense referred to as DHCP snooping is shown in fig. 4.3. In fig. 4.4, the Client and the DHCP Spoofer are considered as untrusted and the Server is trusted. The AP, in the role of DHCP Snooper, is able to detect all DHCP Offers it routes.

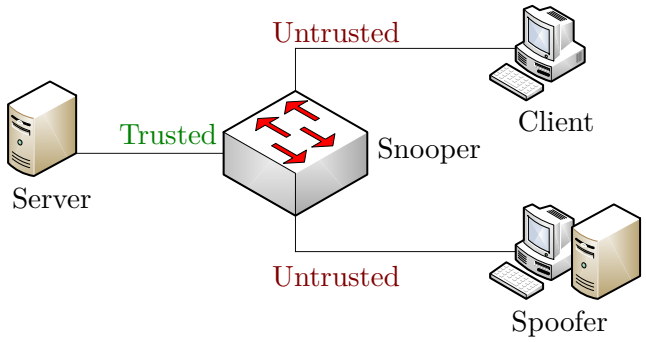


Figure 4.3: Schema of DHCP Snooping

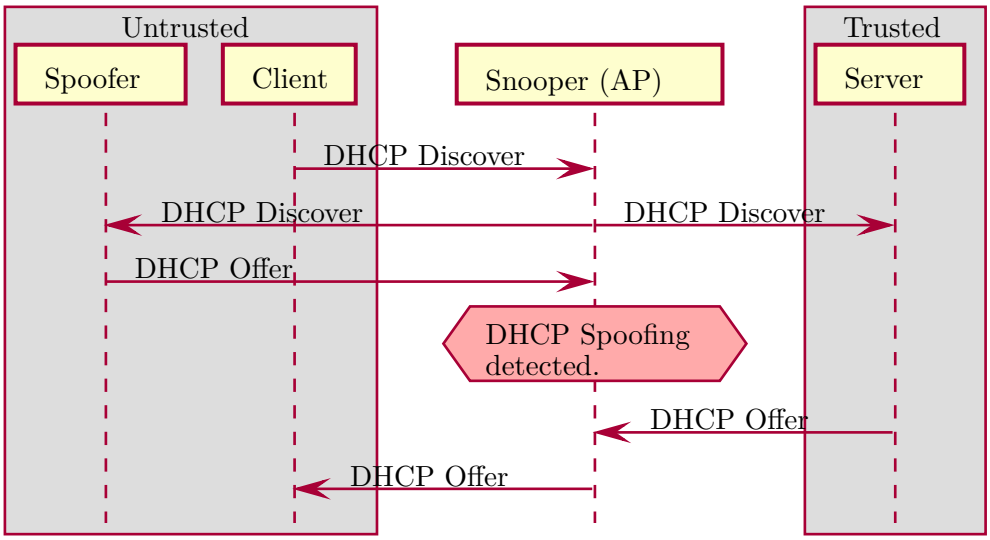


Figure 4.4: Sequence of DHCP Snooping (rest of the communication is omitted)

4.2 ARP Spoofing

A network attack called ARP Spoofing focuses on providing the network devices with fake ARP messages. Description of ARP can be found in section 2.2.

Considering this attack, attacker's possibilities are in persuading the victim that the attacker's MAC address is correctly mapped to some specific IP address. If the attacker's aim is to be in the Man-in-the-Middle position, he is able to persuade the victim about the mapping of default gateway's IP address to the attacker's MAC address. IP address of default gateway in the currently attacked network can be easily obtained for example from transferred DHCP messages (Cisco Systems, Inc. 2016, pp. 56-2).

An attacker can also try to get into the position of Man-in-the-Middle between two devices communicating in the local network. In this case, the attacker would try to persuade each communicating side about the mapping of respective IP addresses to the attacker's MAC address (Cisco Systems, Inc. 2016, pp. 56-2). More information about ARP Spoofing can be found in the literature (Whalen 2001), (Wagner 2001).

Dynamic ARP Inspection

Possible defense against ARP Spoofing attack is analysis of ARP messages transmitted in the network. This method is called Dynamic ARP Inspection (DAI).

As a result of ARP message analysis, given message can be accepted or declined. Network device performing DAI can be configured to distinguish its connections as trusted or untrusted. ARP messages in trusted connections are not analyzed. Analysing device possesses a trusted database of mapping of IP and MAC addresses in the corresponding local network. When the analyzing device receives an ARP message from the untrusted connection, analyzing device can verify the content of ARP message against a trusted mapping database. When a spoofed ARP message with incorrect mapping is detected, ARP message is declined and the incident can be reported to the network administrator (Cisco Systems, Inc. 2016, pp. 56-2).

4.3 IPv6 Neighbor Spoofing

Networks based on IPv6 addressing do not use ARP for IP and MAC address mapping. IPv6 networks use Neighbor Discovery for the similar purpose. Unfortunately, the absence of ARP in IPv6 does not guarantee immunity to the attacks based on a very similar principle as ARP Spoofing.

Neighbor Discovery concerning IPv6 networks is described in section 2.3. In a similar way to ARP Spoofing, attacker can send a specially generated Neighbor Advertisement message to the victim. The main aim of the attacker is, in this case, the same as in the previous attack. The attacker wants to be in the position suitable for the Man-in-the-Middle attack (Stretch 2009). A schema of this attack can be seen in fig. 4.5.

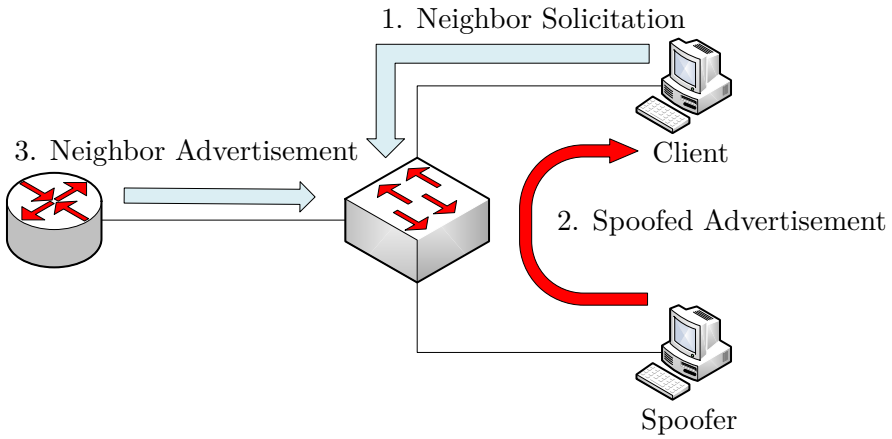


Figure 4.5: Schema of IPv6 Neighbor Spoofing

Neighbor Discovery Inspection

Due to the very similar characteristic of IPv6 Neighbor Spoofing and ARP Spoofing attacks, possible countermeasures are similar as well. Possible defense is Neighbor Discovery Inspection (NDI).

A network device which is performing NDI possesses a trusted database of IP and MAC address mappings. This device verifies information transferred in ND messages against its database. If the device finds an ND message incorrect and spoofed, the message is declined and the incident should be reported (Juniper Networks, Inc. 2014).

4.4 Tools for tampering network topology

In order to change the network topology, the attacker can perform attacks described in previous sections of this chapter. There are already implemented tools and software libraries, which provide functionality for these attacks. Some of these tools are even capable of automated attacks. Available tools are outlined in the following text.

Scapy

Scapy is a program capable of interactive packet manipulation. It is written in the Python language by Philippe Biondi[1]. *Scapy* can be used for capturing, forging and sending packets (Secdev.org 2010).

Main advantage of *Scapy* is the possibility of incorporating its functionality into an automated script, because it is available not only as an interactive program, but also as a Python package. *Scapy* as a Python package is also extensible using add-ons. More information about tool *Scapy* can be found in its documentation (Secdev.org 2010).

dsniff

Collection of network auditing and penetration testing tools called *dsniff* contains several advanced programs, which could be used for tampering network topology. This collection was developed by Dug Song[2]. In the following sections, selected tools from *dsniff* collection are introduced. Detailed information concerning *dsniff* collection is available on its website (Song 2001).

arpspoof

For the realization of ARP Spoofing attack, which is described in section 4.2, it is possible to use the functionality of a tool called *arpspoof*. This software is focused on sending spoofed ARP messages into selected local network. Tool *arpspoof* is a constituent part of dnsiff collection (Song 2001).

Yersinia

The *Yersinia* framework is focused on several weaknesses of network protocols. It can be used to perform a DHCP Spoofing attack, which is described in section 4.1. *Yersinia* offers implementation of DHCP server, which is focused on this attack (Andrés and Barroso 2005a, p. 21). More information about the *Yersinia* framework can be retrieved from its website (Andrés and Barroso 2005b).

[1] URL: http://www.secdev.org/projects/scapy/doc/backmatter.html
[2] URL: https://www.monkey.org/~dugsong/

MITMf

Framework for Man-In-The-Middle attacks (MITMf) offers a wide range of incorporated tools aimed at MitM attack. For the purpose of this thesis, *MITMf* is especially important for included HTTP and DNS servers. It also offers various plugins which can be used during the attack. An available plugin called *Spoof* provides functionality to perform several attacks aimed at traffic redirection such as ARP Spoofing and DHCP Spoofing, which are described in sections 4.1 and 4.2. *MITMf* is written in the Python language and more details about this framework can be found on its repository website (byt3bl33d3r 2014).

4.5 Summary

The chapter Tampering network topology was focused on providing an overview of relevant network attacks, which could be used to establish a network topology suitable for the Man-in-the-Middle attack.

DHCP Spoofing can be used to change the victim's default gateway or DNS. Success of DHCP Spoofing depends on fast delivery of a spoofed DHCP Offer. Possible impact of DHCP Spoofing can be eliminated by DHCP Snooping.

An attack called ARP Spoofing offers a way to impersonate targeted device in the local network. An impersonated MAC address can belong to default gateway, in which case the attacker would redirect all network traffic. ARP Spoofing can be detected by Dynamic ARP Inspection.

In IPv6 networks, IPv6 Neighbor Spoofing can be used for the similar purpose as ARP Spoofing. It could be detected and stopped by Neighbor Discovery Inspection. Section 4.4 serves as an overview of currently available tools, which can be used for realization of the attacks described in sections 4.1 to 4.3.

Chapter 5

Man-in-the-Middle attack

In the following part of this work, the principle of a Man-in-the-Middle attack is described. Concerning the network attack, important prerequisites for its realization and possibilities of exploitation are outlined as well.

Man-in-the-Middle (MitM) refers to the situation, where the attacker's device is located in the network topology between two participants of the communication. The attacker then acts as an intermediary and the network traffic is routed through the attacking device. This state of unauthorized intentionally changed network topology enables the attacker to eavesdrop on passed communication. The attacker is also able to focus on decryption of secured data transfer and on changing the content of passed communication. That means that the attacker can inject special and harmful content. The attacker's prioritized intention is not only to take control over the traffic but also to perform this attack without anyone noticing it.

In the wireless network topology suitable for realization of this attack, the attacker's device acts towards the victim as its default gateway. All the communication routed outside the local network from the victim is sent to the default gateway, in this case to the attacker's device. From the attacker's device, the communication can be further routed to the real default gateway (Callegati, Cerroni, and Ramilli 2009). For the successful execution of this attack, the attacker needs to be connected to the targeted network. Possibilities of connecting to the secured wireless networks are described in sections 3.2 to 3.7. In order to tamper the network topology, an attacker can perform the attacks outlined in chapter 4 using the tools listed in the section 4.4.

5.1 MitM attack to the HTTPS

HTTPS uses asymmetric cryptography with private and public keys to provide secure HTTP communication. If the victim is communicating using HTTPS, successful realization of MitM attack is more difficult.

During communication of web browser on client's device with a web server, these two parties exchange a certificate containing a public key for providing a secure data transfer. MitM attack, in this case, captures transferred certificate and replaces it with a forged one (Callegati, Cerroni, and Ramilli 2009). The forged certificate is at this point a self-signed certificate. Upon reception of the self-signed certificate, victim's web browser can show some warning concerning possible risk. If the victim is not aware of the possible consequences, the victim can accept the certificate. In the case of success, both communi-

cating devices are convinced of secured HTTPS communication, but the attacking device has the ongoing communication available.

5.2 DNS Spoofing

Network attack DNS Spoofing focuses on possibilities of spoofing DNS communication used for resolution of domain names and IP addresses. For the successful realization of this attack, the attacker needs to be able to detect and intercept DNS messages in the network.

The aim of this attack is to direct the victim to different device by providing fake mapping of inquired domain name to a special IP address. The attacker is able to imitate the inquired service by running a similar rogue service on the provided spoofed IP address. If the victim is convinced that the inquired service is genuine, the attacker can then focus on capturing confidential information and credentials. The attacker can also use DNS Spoofing for providing the real service, but with enclosed harmful content. DNS Spoofing can be effectively applied for spoofing fake websites (Prowell, Kraus, and Borkin 2010, p. 112).

If the attacker detects a DNS message, he intercepts it and forges a reply for the victim. The victim receives forged mapping of domain name to IP address and starts communication with the fake device without noticing the attack. The attacker then acts as the inquired service and therefore performs a Man-in-the-Middle attack.

dnsspoof

For the realization of DNS Spoofing, it is possible to use tool *dnsspoof*, which is a part of *dsniff* collection. Collection *dsniff* was introduced in section 4.4 and more information can be found on the respective website (Song 2001).

5.3 Summary

In chapter 5, the Man-in-the-Middle attack was introduced. MitM attack is a dangerous network attack, which grants the attacker ability to eavesdrop and even spoof the victim's communication without victim's knowledge.

Section 5.1 describes the experiments proving that HTTPS communication can be harmed as well. Further described DNS Spoofing in combination with MitM attack can be used for phishing for victim's credentials by impersonation of requested websites and other services.

Chapter 6

Automation of MitM attack

This chapter focuses on the development of the software tool capable of automation of Man-in-the-Middle attack. Different kinds of security approaches used in wireless networks were outlined in chapter Wireless Local Area Networks together with known vulnerabilities. Chapter Significant network technologies introduced currently used technologies. The ways how security flaws in these technologies can be exploited to establish the MitM topology were explained in chapter Tampering network topology. Principle of MitM attack and its severity were outlined in chapter Man-in-the-Middle attack.

Finally, this chapter utilizes previously provided network security knowledge. The implemented software stands on this knowledge and provides automation of the whole process without need for intervention from the user.

Functionality for automated Man-in-the-Middle attack on Wi-Fi networks is provided by implemented Python package named *wifimitm*. The name of the package is derived from *Wi-Fi Machine-in-the-Middle*. Abilities of the *wifimitm* package are utilized in implemented CLI named *wifimitmcli*.

6.1 Design

After studies and research of the appropriate fields from the network security, the research was focused on the available tools for the individual phases of the automated MitM attack. These tools were outlined in sections 3.7 and 4.4 based on their main purpose.

From perspective of the intended functionality of the implemented tool, whole process of MitM attack on wireless networks can be divided into three main phases: *Accessing wireless network*, *Tampering network topology* and *Capturing traffic*. State diagram of the designed phases is in appendix B.

First phase named *Accessing wireless network* consists of several states. At first, the tool needs to scan the wireless networks and find the one selected for attack. If the network is not secured at all, the tool can connect immediately. If the network uses some security algorithm, the tool needs to be able to crack the network password. In the case that the network is secured properly, the tool should try to perform a phishing attack by impersonating the genuine network AP. Aim of this phishing attack is to obtain the network password from its users. Following the successful cracking or phishing, the tool needs to connect to the network. At the point, when the tool is successfully connected to the attacked network, phase *Accessing wireless network* ends.

Considering the phase *Accessing wireless network*, the tools described in section 3.7 can

be utilized. *Airmon-ng* can manage modes of a wireless interface. *Airodump-ng* can be used to scan and detect attacked AP. *Aircrack-ng* together with *aireplay-ng* and *airodump-ng* can be used for cracking WEP OSA, WEP SKA, WPA PSK and WPA2 PSK. The tool *wifiphisher* can be used to perform impersonation and phishing. Connection to the wireless network can be established by *netctl*[1], which is a CLI tool focused on configuration and management of network connections using profiles.

Phases *Tampering network topology* and *Capturing traffic* are concurrent, as shown in the appendix B. These two phases form a MitM attack. MitM attack could possibly consist of spoofing victim's communication, as described in chapter 5. During *Tampering network topology*, the tool needs to be able to change network topology into the one suitable for MitM attack. The tool also needs to continuously work on keeping the network STAs persuaded that the spoofed topology is the correct one. During the phase *Capturing traffic*, the tool needs to be able to save network traffic into a capture file.

The tools from section 4.4 can be used during the *Tampering network topology* phase. *Framework for Man-In-The-Middle attacks* with its *Spoof* plugin can be used to change the network topology. *Capturing traffic* can be done by the tool *dumpcap*[2], which is part of the *Wireshark*[3] distribution.

During this part of the work on the automated tool, the available tools were examined. Several experiments concerning wireless network security and tampering network topology were carried out. Behaviour, usage and success rate of individual tools as well as possibilities of controlling them by the implemented tool were analyzed. The tools selected for individual tasks of the automated MitM attack were chosen based on performed manual experiments.

6.2 Implementation

After completion of design of main characteristic and selection of tools for utilization by the automated tool, implementation part of the work on the tool began. The implemented tool is intended to run on *Arch Linux*[4]. This distribution was selected, because its very lightweight and flexible. The user has possibilities of better control over the behaviour of the operating system. This distribution was used for main development and testing as well.

According to the thesis assignment, Python 3.5 was selected as main implementation language for the automated tool. Bash was selected as implementation language for supporting tasks, e.g., installation of requirements and software wrappers.

Package structure

After finishing the development version *release-0.1*, which represented the first prototype of the automated tool, more attention was paid to improving the architecture of the automated tool. In the field of software development, very important aspects of the final software product are, among other things, its re-usability, possibility and ease of modification and ways of incorporating its functionality to other software products which would build on top of it. This is especially important in the area of automated tools.

Architecture of the tool was therefore changed and improved. Main functionality of the individual tasks concerning the MitM atack on Wi-Fi networks was separated from the

[1]URL: https://www.archlinux.org/packages/core/any/netctl/
[2]URL: https://www.wireshark.org/docs/man-pages/dumpcap.html
[3]URL: https://www.wireshark.org/
[4]URL: https://www.archlinux.org/

user interface. This way, a Python package providing attack functionality was developed. User interface for the implemented tool was developed as a script, which serves as an entry point for the package. Using this approach, it is possible to develop more than one user interface, e.g., Command Line Interface, Text User Interface, Graphical User Interface. These individual interfaces could be separated from each other and even their dependencies for required technologies could be separated.

Functionality implemented in the *wifimitm* package could be directly incorporated to other software products based on Python language. This way the package would work as a software library. An attack strategy could be modified as well by changing the way of interaction with the package's content. Structure of the implemented automated tool is in appendix C.

Another advantage of implementing the functionality as a Python package comes in the moment of software distribution. With the use of Python's *setuptools*, a package can declare its metadata, requirements, resources, entry points and other characteristics. A package coupled with a file *setup.py* can be then easily distributed and installed including its requirements. This approach also allows convenient usage of additional resources, which are distributed with a package. Tools incorporating a package can then declare the package as a dependency.

The *wifimitm* package consists of following modules: *access*, *capture*, *common*, *impersonation*, *model*, *requirements*, *topology*, *updatableProcess*, *wep*, *wpa2*. The package further contains folder for its resources and subpackage *tests* containing implemented unit tests. Package's modules contain their individual logger objects used for logging executed steps at various logging levels.

Due to the fact, that the *wifimitm* needs to interact with other software tools, it requires these software tools for its successful execution. Nature of the intended operations during the individual steps of the attack also requires the tool to be executed with the root privileges. The *requirements* module focuses on managing execution requirements and their check during start of the tool. This module declares abstract base class for a generic requirement and two more specific classes, one for executable software tool requirement and the other for UID requirement. Instances of individual requirements are stored and can be easily checked.

The *access* module focuses on accessing wireless network. It uses modules *wep* and *wpa2*, which implement attacks and cracking based on the used security algorithm. The *access* module offers an automated process of cracking selected wireless network. The *wep* module is capable of fake authentication with the AP, ARP replay attack (to speed up gathering of IVs) and cracking the key based on IVs. In the case of WPA2 secured network, the *wpa2* module is able to perform a dictionary attack, personalize used dictionary and verify a password obtained by phishing. Verification of the password and dictionary attacks are done with previously captured handshake. The *common* module contains functionality which could be used in various parts of the process for scanning and capturing wireless communication in monitor mode. The *common* module also offers a way to deauthenticate STAs from selected AP.

If a dictionary attack against a correctly secured network fails, the *impersonation* module is able to manage a phishing attack by impersonation. The *topology* module can be used to change network topology. It provides functionality for ARP Spoofing. The *capture* module focuses on capturing network traffic. It is intended to be used after the tool is successfully connected to the attacked network and network topology was successfully changed into the one suitable for MitM attack.

Above described modules of the *wifimitm* package need to interact with STAs, APs and wireless network interfaces. The *model* module supports the rest of the package with implemented classes representing the network and interfaces for better interaction. This module also provides functionality to save useful files that were obtained during attacks. More information about saved files is in section Attack data.

Incorporation of tools

As noted previously, the implemented tool needs to be able to interact with many other software tools in order to automate the attack process. Incorporated tools communicate using Standard output stream (stdout), Standard error stream (stderr) and optionally using generated files. *Wifimitm* needs to continuously analyze all these outputs to be aware of current state of the incorporated tool. Information contained in the output can be a summary of current progress, a notification that some event occurred or a result of an intended action.

To meet requirements for efficient incorporation of other tools so that the *wifimitm* package could interact with them, the *updatableProcess* module was developed. This module contains an abstract base class *UpdatableProcess*, which inherits from *Popen*[5] class implemented in the *subprocess* module of *The Python Standard Library*. Individual incorporated tools have dedicated classes inherited from the *UpdatableProcess* which are used for managing these tools from *wifimitm*.

When a process is spawned, using an instance of class inherited from *UpdatableProcess*, it is assigned a temporary directory for its outputs. The running process is continuously writing to stdout and stderr. The outputs are periodically analyzed.

UpdatableProcess offers a functionality to be used as a context manager. Upon entering the context, a process is spawned. On exit from the context, the process is waited for and a cleanup is done. A running process can be stopped and cleaned using available methods of a corresponding class. Cleanup of the process' outputs including opened files and created directories can be done using the *cleanup* method. In order to automatically perform a cleanup, *UpdatableProcess* uses a destructor method and a finalizer object, implemented in the *weakref*[6] module of *The Python Standard Library*.

Classes inherited from *UpdatableProcess* can implement a signalization of process' state, concerning process' purpose, using a Finite State Machine (FSM). Process' output can include notifications of events. Upon detection of such event, flags can be set. Some processes also output summary information, which can be used to update statistics. Continuously updated information about the process can therefore consist of state, flags and statistics. Abstract base class *UpdatableProcess* contains an abstract method *update*, which is intended to implement a logic of an analysis of the process' outputs. The analysis can result in modification of process' state, flags and statistics.

Attack data

Various attacks executed against the selected AP require some information to be captured first. ARP request replay attack on WEP secured networks requires an ARP request to be captured in order to start an attacking procedure. Fake authentication in WEP SKA secured network requires PRGA XOR obtained from a detected authentication. Dictionary

[5]URL: https://docs.python.org/3/library/subprocess.html#subprocess.Popen
[6]URL: https://docs.python.org/3/library/weakref.html#weakref.finalize

attack against WPA PSK and WPA2 PSK secured networks requires a captured handshake. Finally, for successful connection to a network, a correct key is required.

When the required information is obtained, it can be saved for a later usage to speed up following or repetitive attacks. These data are implicitly stored as files inside user's home directory in `.wifimitm/` directory. Data from successful attacks could be even shared between users of the implemented tool. In possible future development of the tool, the data could be stored in a shared database and accessed by users.

Personalized dictionaries

As described in section 3.6, weaknesses in default network passwords could be exploited to improve dictionary attacks against WPA PSK and WPA2 PSK security algorithms.

The implemented tool incorporates *upc_keys*[7] for generation of possible default passwords if the selected network matches the criteria. The *upc_keys* tool generates passwords, which are transferred to the cracking tool using pipes. With this approach, the implemented tool could be further improved for example to support localized dictionaries.

6.3 Installation

Based on the previously described facts, the implemented automated tool depends on several other tools, which are being controlled. *Wifimitm* has to be able to start the required tools, therefore they have to be available on a user's system. The *wifimitm* package itself can be automatically installed by the package's *setup.py*. After the installation, the implemented automated tool can be started using its CLI named *wifimitmcli*. The rest of software dependencies can be satisfied by installation of required tools. For convenient setup of the implemented tool, a *Makefile* and several installation scripts and wrappers have been developed.

MITMf has a number of dependencies, therefore it is highly recommended to use MITMf inside a virtual environment as stated in its installation guide[8]. MITMf could be installed using the package[9] available on Arch User Repository (AUR), but unfortunately this package does not utilize the virtual environment. An installation script *MITMf_install.sh* is able to install MITMf, including its dependencies. This script also creates a virtual environment dedicated to MITMf. An implemented wrapper script is used to automate activation and deactivation of the virtual environment before and after running MITMf. After installation, MITMf can be easily run encapsulated in its virtual environment.

Wifiphisher is available in form of an AUR package[10], but this package is not suitable for correct installation, because currently (May 2016), it is not updated to the changes in the repository structure of *wifiphisher*. An implemented installation script *wifiphisher_install.sh* is able to create a dedicated virtual environment and install *wifiphisher*. Convenient usage of *wifiphisher* installed inside its virtual environment is achieved by a wrapper similar to the one for MITMf. Due to the fact that some changes in *wifiphisher*'s source code were implemented, the installation script also applies a software patch to the installed *wifiphisher*.

Tool *upc_keys* is implemented in the C language and therefore it is compiled during installation. Compiled *upc_keys* and the executable wrappers for MITMf and *wifiphisher*,

[7] URL: https://haxx.in/upc-wifi/
[8] URL: https://github.com/byt3bl33d3r/MITMf/wiki/Installation
[9] URL: https://aur.archlinux.org/packages/mitmf-git/
[10] URL: https://aur.archlinux.org/packages/wifiphisher/

which are described above, are linked from the `/usr/bin/` directory after the installation. The required tools are installed by their installation scripts to the `/opt/` directory. Installation of all the requirements can be started by *requirements_install.sh* script or *Makefile*. A usage of implemented *Makefile*, which can be used for a convenient installation, is shown in table 6.1.

Table 6.1: A usage of *Makefile*

Command	**Description**
`make requirements`	Install requirements.
`make install`	Install the *wifimitm* package and the *wifimitmcli* tool.
`make man`	Install a manual page of *wifimitmcli*.
`make`, `make all`	Install requirements, the package, the tool and the manual page.

Hardware requirements

Due to the nature of specific steps of the attack, a special hardware equipment is required. During the scanning and capturing of network traffic without being connected to the network, an attacking device needs a wireless network interface in monitor mode. For sending special forged packets, the wireless network interface also needs to be capable of packet injection. In order to be able to perform a phishing attack, a second wireless interface capable of master (AP) mode has to be available.

The user can check whether his hardware is capable of packet injection using the *aireplay-ng* tool executed as `aireplay-ng --test <replay interface>`. Managing monitor mode of interface is possible with the *airmon-ng* tool.

6.4 Usage

As described in section 6.3, after the installation the CLI can be started via *wifimitmcli*. During *wifimitmcli*'s run, usual output information is written to stdout, notifications concerning errors are written to stderr. *Wifimitmcli* saves and loads attack data from the `~/.wifimitm/` directory.

According to the fact that *wifimitmcli* is an automated tool, it does not expect any input from a user during its progress. The user can control behaviour of *wifimitmcli* by program arguments provided at start of *wifimitmcli*.

This way, *wifimitmcli* does not even have to be started manually by user, but it could be a part of other scripts. Table 6.2 shows an overview of program arguments of *wifimitmcli* tool. The synopsis of *wifimitmcli*'s arguments is specified as follows: `wifimitmcli [-h] [-v] [-ll <level>] [-p] [-cf FILE] <ssid> <interface>`.

As seen from the synopsis shown above, `<ssid>` and `<interface>` arguments are mandatory to start *wifimitmcli*. In the case that provided arguments are not correct, an appropriate error message and the synopsis is shown and the program terminates immediately after the arguments check. For more information concerning usage of *wifimitmcli*, a user can start the tool with `-h` or `--help` argument, which results in showing a help page. More detailed information about *wifimitmcli* can be found on its installed manual page. Further information can also be found in *README.md*, a file distributed with the source code.

Table 6.2: Program arguments of *wifimitmcli*

Argument	**Description**
`-h`, `--help`	Show help message and exit.
`-v`, `--version`	Show program's version number and exit.
`-ll <level>`, `--logging-level <level>`	Select logging level (choices: *disabled*, *critical*, *error*, *warning*, *info*, *debug*).
`-p`, `--phishing`	Enable phishing attack if dictionary attack fails.
`-cf FILE`, `--capture-file FILE`	Capture network traffic to provided file.
`<ssid>`	Attack network with provided SSID.
`<interface>`	Use provided wireless network interface for attack.

The implemented Python package *wifimitm* provides a functionality to log performed actions using Python's logging[11] module. Individual modules contained in the *wifimitm* package posses their own logger objects. The implemented *wifimitmcli* tool uses its logger as well. This approach makes it possible for *wifimitmcli* to control all noted loggers. Level of logging for the loggers can be set at start of *wifimitmcli* as a program argument.

Upon termination of the *wifimitmcli* tool, appropriate exit code indicating the result is returned. Some of the implemented exit codes are inspired by *sysexits*[12]. Exit codes of the implemented automated tool are shown in table 6.3

Table 6.3: Exit codes of *wifimitmcli*

Value	**Name**	**Description**
0	EX_OK	Program terminated successfully.
2	ARGUMENTS	Incorrect or missing arguments provided.
69	EX_UNAVAILABLE	Required program or file does not exist.
77	EX_NOPERM	Permission denied.
79	TARGET_AP_NOT_FOUND	Target AP was not found during scan.
80	NOT_IN_ANY_DICTIONARY	WPA/WPA2 passphrase was not found in any available dictionary.
81	PHISHING_INCORRECT_PSK	WPA/WPA2 passphrase obtained from phishing attack is incorrect.
82	SUBPROCESS_ERROR	Failure in subprocess occured.
130	KEYBOARD_INTERRUPT	Program received SIGINT.

6.5 Testing

During development, a functionality of the *wifimitmcli* tool and the *wifimitm* package was periodically tested. After the implementation part of the work on this thesis was finished, several experiments were carried out in order to test the final implemented software product.

The author was working on this research during his studies at Faculty of Information and Communication Technology, University of Malta, in 2016. Unfortunately, no wireless network laboratory was available for experiments during the author's stay at University

[11] URL: https://docs.python.org/3/library/logging.html
[12] URL: http://linux.die.net/include/sysexits.h

of Malta. Due to this fact, the testing experiments were done with a limited range of equipment that was available. A more complex testing is going to be performed at Faculty of Information Technology, Brno University of Technology.

Before the individual experiments, a testing network infrastructure was set up. The network consisted of one STA and one AP connected to the Internet. A scheme of the network used for the experiments is shown in appendix D. The STA was correctly connected with the AP and it was successfully communicating with the Internet. The implemented *wifimitmcli* tool was then started with appropriate program arguments to automatically attack the network. The test was considered successful, if the *wifimitmcli* was able to capture network traffic according to the concept of MitM. For the test to be correct, no intervention (help) from the attacker was allowed during the attack performed by *wifimitmcli*. Summary of experiments is shown in appendix E.

Chapter 7

Conclusion

The main goal of this thesis was to implement a tool that would be able to automate all the necessary steps to perform the Man-in-the-Middle attack on Wireless Local Area Networks. The author searched for and analyzed a range of tools and methods focused on penetration testing, communication sniffing and spoofing, password cracking and hacking in general. In order to be able to design, implement and test the tool capable of such attack, knowledge of different widespread security approaches was essential. Specialized tools focused on exploiting individual weaknesses in security algorithms currently used in WLANs are already available. There are also specialized tools focused on individual steps of the MitM attack. The author further focused on possibilities of the MitM attack even in the case that given WLAN is secured properly. Therefore, methods and tools for impersonation and phishing were researched and analysed.

Based on the acquired knowledge, referenced researches and practical experience from manual experiments, the author was able to create an attack strategy. The strategy was composed of appropriate set of available tools. The strategy is then able to select individual steps which are the most suitable for a successful MitM attack.

The author's work and research resulted in a development of the *wifimitm* package implemented in the Python language. This package serves as a software library which provides functionality for automated MitM attack on WLANs. The implemented package is therefore beneficial for others due to the fact that it can be easily incorporated into other tools. The product of this research is also a tool which incorporates the functionality of the *wifimitm* package. This tool named *wifimitmcli* manages the individual steps of automated MitM attack and serves as a Command Line Interface. The implemented software comes with a range of additions for a convenient usage, e.g., requirements installation scripts, a requirements check, a Python package setup with *wifimitmcli* installation, *README.md* file and a manual page.

The *wifimitmcli* tool, and therefore *wifimitm* as well, was tested during experiments with an available set of equipment. As the results show, the implemented software product is able to successfully perform an automated MitM attack on WLANs. More detailed testing, in order to obtain a better proved conclusion, is going to be performed in a proper wireless network laboratory.

A possible future progress of the project, which started by this thesis, is considered especially in improving the success rate of the automated strategy. In the future iterations of the development, the product could focus on exploiting the weaknesses of widely used Wi-Fi Protected Setup. Concerning the current state of the product, it does not focus on enterprise WLANs, which could be vulnerable as well. Further development should also

consider countermeasures of Intrusion Prevention Systems (IPSs).

At this point, the attack can be done automatically using the implemented CLI. Depending on a target group of users of the product, future development could also consider implementation of various TUIs and GUIs. Implementation of various interfaces is possible due to the fact that the product is implemented as a package providing all the required functionality.

An advantage of automation of some process can be the fact, that the user has less work to do and that deep understanding of the individual steps is not required for the usage. But most importantly, the automated process is able to operate completely without a presence of a human. The research product could be even used on a device of miniature dimensions, as long as the device is able to satisfy the software and hardware requirements. This way, an automated probe could be constructed. An operation would then consist of deploying the probe device and its activation. The probe would then be able to perform an automated attack and collect important data without any further human intervention.

This research and its products could find a good utilization in combination with other security researches carried out at the Brno University of Technology, Faculty of Information Technology. It can serve in an investigation done by forensic researchers (Pluskal et al. 2015). A software capable of automated MitM attack on WLANs can also be used to improve the security of networks by automatically detecting their vulnerabilities. This way, *wifimitmcli* can be considered an automated penetration testing tool.

Finally, popularization of the fact, that such severe attacks can be successfully automated, should be used to raise the public awareness about the information security. In these days, this issue involves almost every one of us.

Bibliography

[1] Alfredo Andrés and David Barroso. *Yersinia - Framework for layer 2 attack*. Online. 2005. URL: http://blackhat.com/presentations/bh-europe-05/BH_EU_05-Berrueta_Andres/BH_EU_05_Berrueta_Andres.pdf (visited on 01/24/2016).

[2] Alfredo Andrés and David Barroso. *Yersinia.net*. Online. Yersinia.net, 2005. URL: http://www.yersinia.net (visited on 01/24/2016).

[3] byt3bl33d3r. *Framework for Man-In-The-Middle attacks*. Online. July 2014. URL: https://github.com/byt3bl33d3r/MITMf (visited on 04/26/2016).

[4] F. Callegati, W. Cerroni, and M. Ramilli. "Man-in-the-Middle Attack to the HTTPS Protocol". In: *Security Privacy, IEEE* (Jan. 2009), pp. 78–81. ISSN: 1540-7993. DOI: 10.1109/MSP.2009.12.

[5] Cisco Systems, Inc. *Catalyst 6500 Release 12.2SX Software Configuration Guide*. Online. Cisco Systems, Inc. URL: http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book.html (visited on 01/24/2016).

[6] Cisco Systems, Inc. *IPv6 Configuration Guide, Cisco IOS Release 15.2M&T*. Online. Cisco Systems, Inc., 2012. URL: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/configuration/15-2mt/ipv6-15-2mt-book.html (visited on 01/24/2016).

[7] CTU.cz. *Využívání vymezených rádiových kmitočtů | Český telekomunikační úřad*. Online. ČTÚ, 2016. URL: https://www.ctu.cz/vyuzivani-vymezenych-radiovych-kmitoctu (visited on 01/22/2016).

[8] Thomas d'Otreppe. *Aircrack-ng*. Online. 2016. URL: http://www.aircrack-ng.org/ (visited on 04/27/2016).

[9] R. Droms. *Dynamic Host Configuration Protocol*. Tech. rep. 2131. Updated by RFCs 3396, 4361, 5494, 6842. Mar. 1997. URL: http://www.ietf.org/rfc/rfc2131.txt.

[10] S. Duangphasuk, S. Kungpisdan, and S. Hankla. "Design and implementation of improved security protocols for DHCP using digital certificates". In: *2011 17th IEEE International Conference on Networks*. Dec. 2011, pp. 287–292. DOI: 10.1109/ICON.2011.6168490.

[11] Wi-fi.org. *Certification | Wi-Fi Alliance*. Online. Wi-Fi Alliance, 2016. URL: http://www.wi-fi.org/certification (visited on 01/22/2016).

[12] Scott Fluhrer, Itsik Mantin, and Adi Shamir. "Weaknesses in the Key Scheduling Algorithm of RC4". English. In: *Selected Areas in Cryptography*. Ed. by Serge Vaudenay and AmrM. Youssef. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2001, pp. 1–24. ISBN: 978-3-540-43066-7. DOI: 10.1007/3-540-45537-X_1. URL: http://dx.doi.org/10.1007/3-540-45537-X_1.

[13] F. Halsall. *Computer Networking and the Internet*. Online. Addison-Wesley, 2005, pp. 1–803. ISBN: 9780321263582. URL: https://books.google.cz/books?id=QadX5XErZ9IC (visited on 01/22/2016).

[14] Craig Heffner. *Cracking WPA in 10 Hours or Less – /dev/ttyS0*. Online. 2011. URL: http://www.devttys0.com/2011/12/cracking-wpa-in-10-hours-or-less/ (visited on 04/26/2016).

[15] IEEE-SA. "IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications". In: *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)* (Mar. 2012), pp. 1–2793. DOI: 10.1109/IEEESTD.2012.6178212.

[16] IEEE-SA. "IEEE Standard for information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements". In: *IEEE Std 802.11i-2004* (July 2004), pp. 1–190. DOI: 10.1109/IEEESTD.2004.94585.

[17] Martin Jelínek. "Bezpečnost bezdrátových počítačových sítí". MA thesis. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. URL: http://hdl.handle.net/11012/21018.

[18] Juniper Networks, Inc. *Understanding IPv6 Neighbor Discovery Inspection*. Online. Juniper Networks, Inc., Apr. 2014. URL: http://www.juniper.net/documentation/en_US/junos13.2/topics/concept/port-security-nd-inspection.html (visited on 01/24/2016).

[19] Vishal Kumkar et al. "Vulnerabilities of Wireless Security protocols (WEP and WPA2)". In: *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)* 1.2 (2012), pp. 34–38. ISSN: 22781323. URL: http://ijarcet.org/wp-content/uploads/IJARCET-VOL-1-ISSUE-2-34-38.pdf.

[20] Y. Liu, Z. Jin, and Y. Wang. "Survey on Security Scheme and Attacking Methods of WPA/WPA2". In: *2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*. Sept. 2010, pp. 1–4. DOI: 10.1109/WICOM.2010.5601275.

[21] D. Plummer. *Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware*. Tech. rep. 826. Updated by RFCs 5227, 5494. Nov. 1982. URL: http://www.ietf.org/rfc/rfc826.txt.

[22] Jan Pluskal et al. "Netfox Detective: A tool for advanced network forensics analysis". In: *Proceedings of Security and Protection of Information (SPI) 2015*. Brno, CZ: Brno University of Defence, 2015, pp. 147–163. ISBN: 978-80-7231-997-8. URL: http://www.fit.vutbr.cz/research/view_pub.php?id=10863.

[23] Stacy Prowell, Rob Kraus, and Mike Borkin. “CHAPTER 6 - Man-in-the-Middle”. In: *Seven Deadliest Network Attacks*. Ed. by Stacy Prowell, Rob Kraus, and Mike Borkin. Boston: Syngress, 2010, pp. 101–120. ISBN: 978-1-59749-549-3. DOI: http://dx.doi.org/10.1016/B978-1-59749-549-3.00006-7. URL: http://www.sciencedirect.com/science/article/pii/B9781597495493000067.

[24] Pieter Robyns. “Wireless Network Privacy”. MA thesis. Hasselt: Hasselt University, 2014. URL: http://hdl.handle.net/1942/17516.

[25] Secdev.org. *Welcome to Scapy's documentation! — Scapy v2.1.1-dev documentation*. Online. Secdev.org, Apr. 2010. URL: http://www.secdev.org/projects/scapy/doc/ (visited on 01/24/2016).

[26] Dug Song. *dsniff*. Online. Monkey.org, Dec. 2001. URL: http://www.monkey.org/~dugsong/dsniff/ (visited on 01/24/2016).

[27] Jeremy Stretch. *IPv6 neighbor spoofing - PacketLife.net*. Online. PacketLife.net, Feb. 2009. URL: http://packetlife.net/blog/2009/feb/2/ipv6-neighbor-spoofing/ (visited on 01/24/2016).

[28] Erik Tews, Ralf-Philipp Weinmann, and Andrei Pyshkin. “Breaking 104 Bit WEP in Less Than 60 Seconds”. English. In: *Information Security Applications*. Ed. by Sehun Kim, Moti Yung, and Hyung-Woo Lee. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 188–202. ISBN: 978-3-540-77534-8. DOI: 10.1007/978-3-540-77535-5_14. URL: http://dx.doi.org/10.1007/978-3-540-77535-5_14.

[29] Stefan Viehböck. *Brute forcing Wi-Fi protected setup*. Online. 2011. URL: https://sviehb.files.wordpress.com/2011/12/viehboeck_wps.pdf (visited on 04/26/2016).

[30] Stefan Viehböck. “Wi-Fi Protected Setup PIN brute force vulnerability”. In: *.braindump - RE and stuff* (2011). Online. URL: https://sviehb.wordpress.com/2011/12/27/wi-fi-protected-setup-pin-brute-force-vulnerability/ (visited on 04/26/2016).

[31] Robert Wagner. “Address resolution protocol spoofing and man-in-the-middle attacks”. In: *The SANS Institute* (Aug. 2001). Online. URL: https://www.sans.org/reading-room/whitepapers/threats/address-resolution-protocol-spoofing-man-in-the-middle-attacks-474 (visited on 04/26/2016).

[32] Sean Whalen. *An Introduction to Arp Spoofing*. Online. Apr. 2001. URL: https://dl.packetstormsecurity.net/papers/protocols/intro_to_arp_spoofing.pdf (visited on 04/26/2016).

Acronyms

AP Access Point.

ARP Address Resolution Protocol.

AUR Arch User Repository.

CLI Command Line Interface.

CRC Cyclic Redundancy Check.

DAI Dynamic ARP Inspection.

DHCP Dynamic Host Configuration Protocol.

DNS Domain Name System.

FSM Finite State Machine.

GPU Graphics Processing Unit.

GUI Graphical User Interface.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

ICMPv6 Internet Control Message Protocol version 6.

IP Internet Protocol.

IPS Intrusion Prevention System.

IPv4 Internet Protocol version 4.

IPv6 Internet Protocol version 6.

IV Initialization Vector.

MAC Media Access Control.

MitM Man-in-the-Middle.

MITMf Framework for Man-In-The-Middle attacks.

NA Neighbor Advertisement.

ND Neighbor Discovery.

NDI Neighbor Discovery Inspection.

NS Neighbor Solicitation.

OSA Open System Authentication.

PIN Personal Identification Number.

PMK Pairwise Master Key.

PRGA Pseudo Random Generation Algorithm.

PSK Pre-Shared Key.

RA Router Advertisement.

RC4 Rivest Cipher 4.

SKA Shared Key Authentication.

SSID Service Set Identifier.

STA Station.

stderr Standard error stream.

stdout Standard output stream.

TKIP Temporal Key Integrity Protocol.

TUI Text User Interface.

UID User Identifier.

WEP Wired Equivalent Privacy.

WLAN Wireless Local Area Network.

WPA Wi-Fi Protected Access.

WPA2 Wi-Fi Protected Access II.

WPS Wi-Fi Protected Setup.

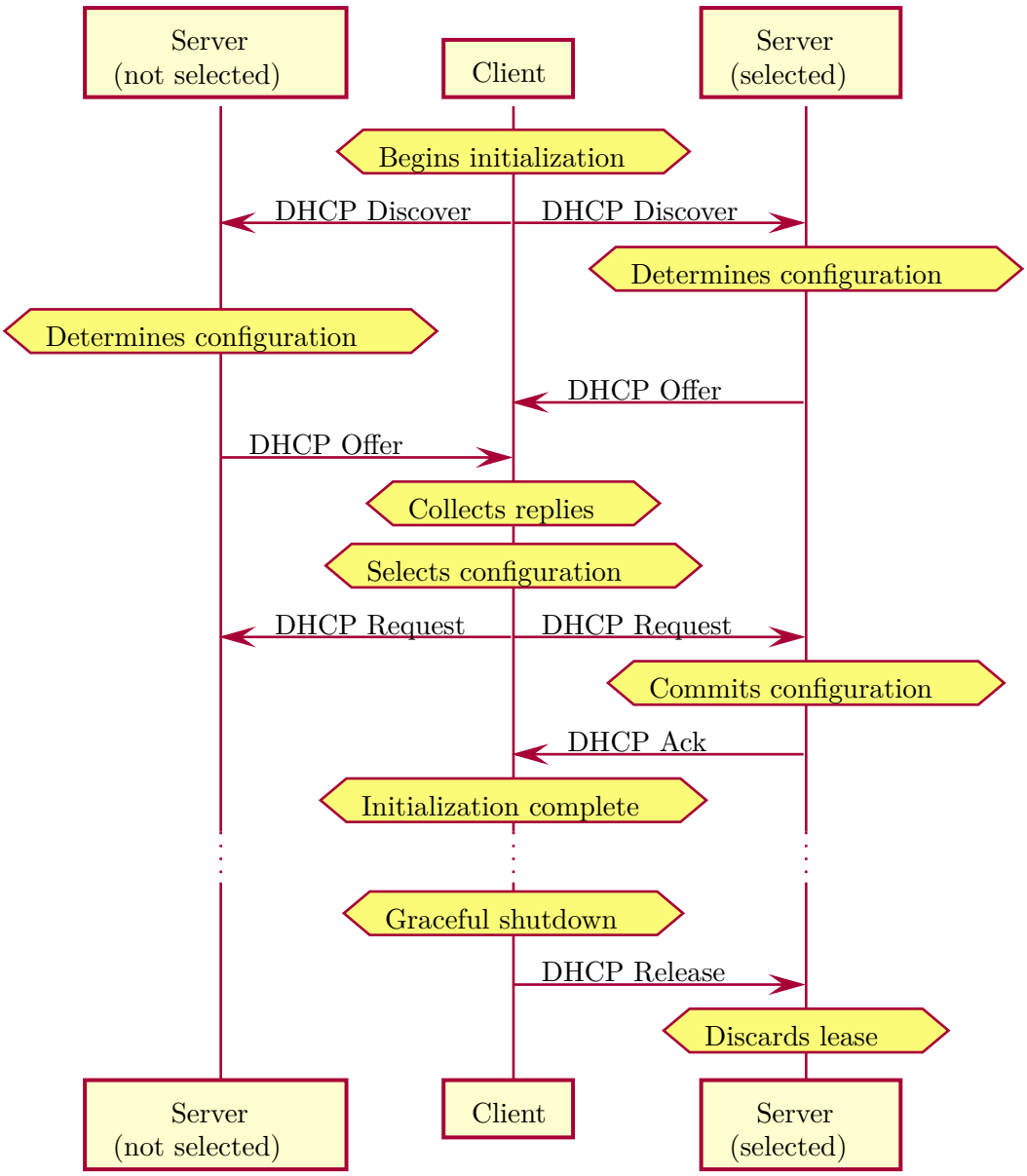
XOR Exclusive Disjunction.

Appendices

List of Appendices

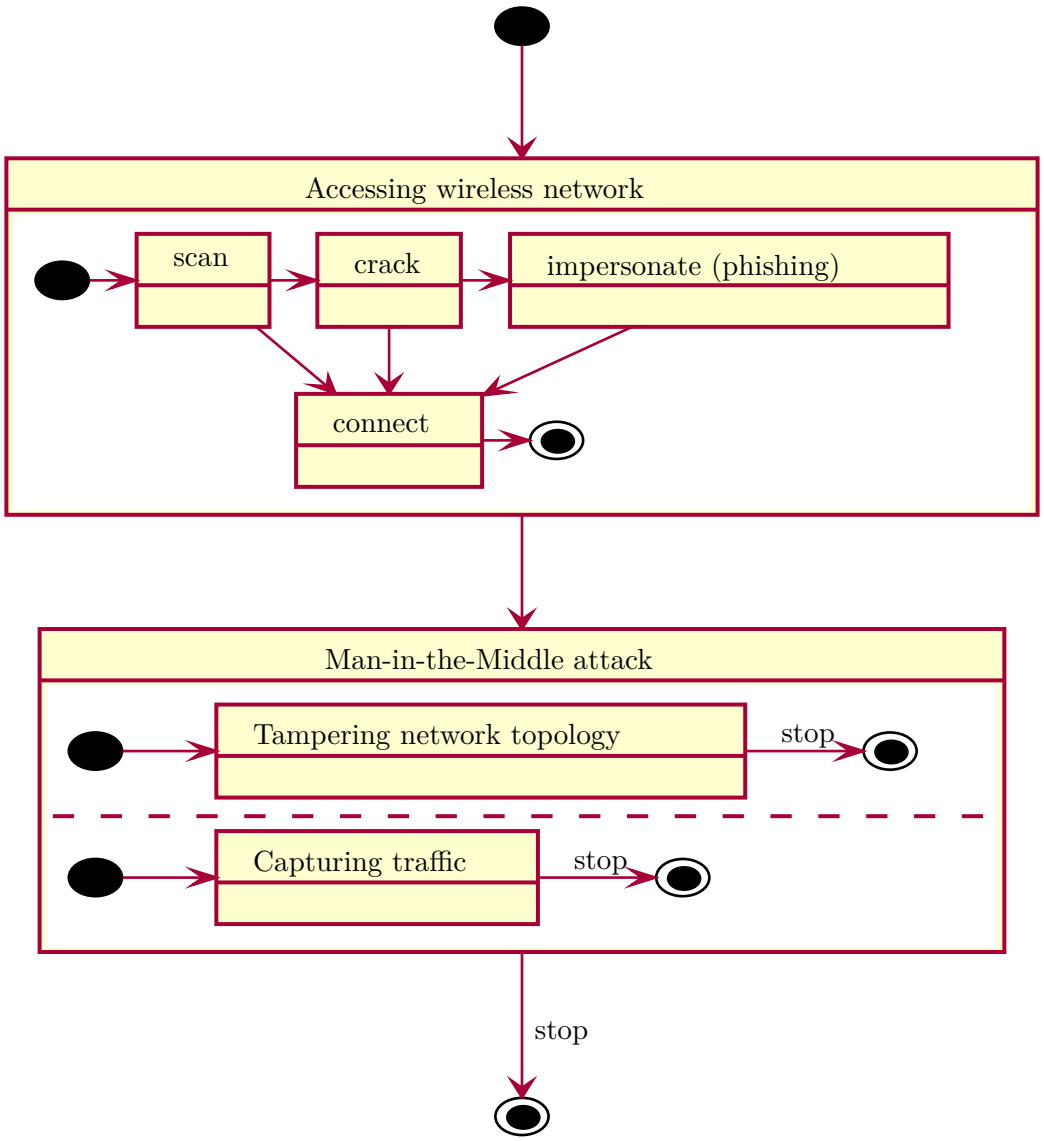
Appendix A

DHCP sequence



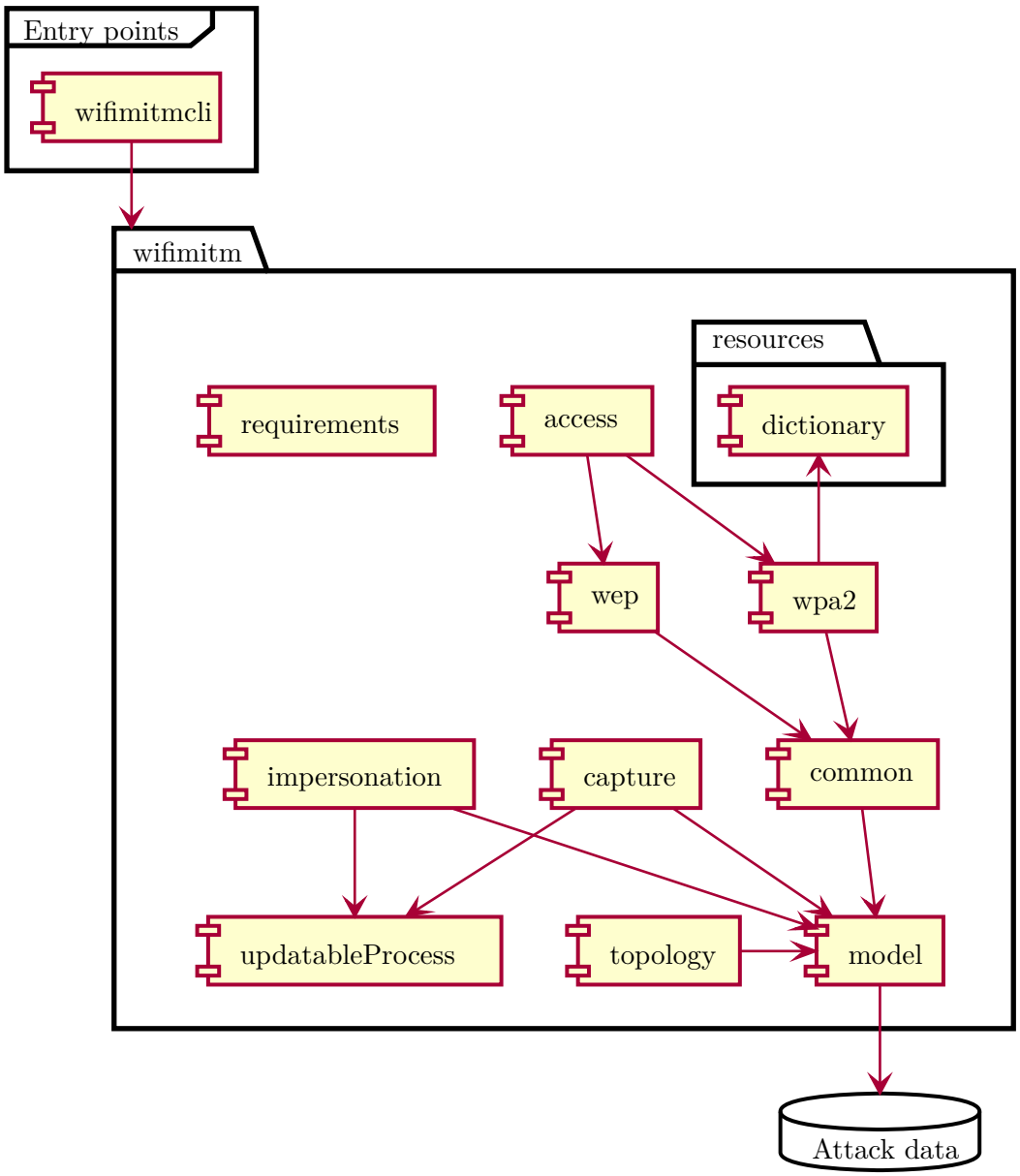
Appendix B

Phases of MitM attack on WLANs



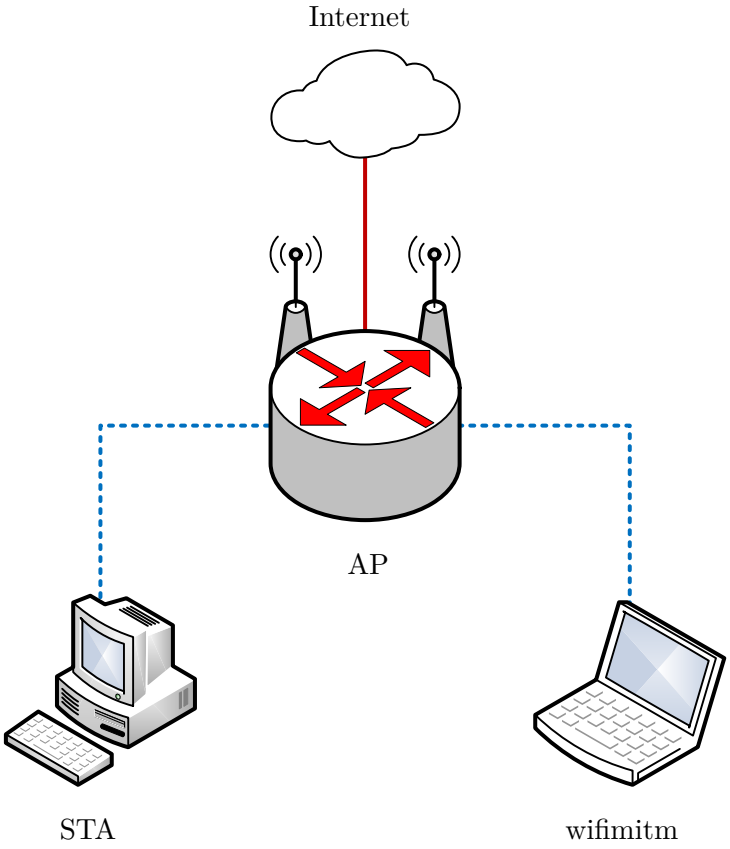
Appendix C

Structure of implemented tool



Appendix D

Schema of experiments



Appendix E

Summary of experiments

Results of experiments (table E.1) show, that open networks can be very easily attacked. WEP OSA and WEP SKA secured networks can be successfully attacked even if they use strong password. WPA PSK and WPA2 PSK secured networks suffer from weak passwords, default passwords and mistakes of users.

Table E.1: Summary of *wifimitmcli's* automated attacks.

Security	PSK	SSID	AP	STA	Result
open		test-open	TP-LINK, TL-WR841N, 3.16.9 Build 150310 Rel.55295n	HTC Desire 500, Android 4.1.2	Success
open		test-open	TP-LINK, TL-WR841N, 3.16.9 Build 150310 Rel.55295n	Lenovo IdeaPad Yoga 2, Arch Linux 4.4.1	Success
WEP OSA	`A_b#1`	test-wep-osa	TP-LINK, TL-WR841N, 3.16.9 Build 150310 Rel.55295n	HTC Desire 500, Android 4.1.2	Success
WEP OSA	`A_b#1`	test-wep-osa	TP-LINK, TL-WR841N, 3.16.9 Build 150310 Rel.55295n	Meizu m2 note, Android 5.1	Success
WEP OSA	`A_b#1`	test-wep-osa	TP-LINK, TL-WR841N, 3.16.9 Build 150310 Rel.55295n	Lenovo IdeaPad Yoga 2, Arch Linux 4.4.1	Success
WEP SKA	`A_b#1`	test-wep-ska	TP-LINK, TL-WR841N, 3.16.9 Build 150310 Rel.55295n	HTC Desire 500, Android 4.1.2	Success

Continued on next page.

Table E.1 (continued)

Security	PSK	SSID	AP	STA	Result
WEP SKA	`A_b#1`	test-wep-ska	TP-LINK, TL-WR841N, 3.16.9 Build 150310 Rel.55295n	Meizu m2 note, Android 5.1	Success
WEP SKA	`A_b#1`	test-wep-ska	TP-LINK, TL-WR841N, 3.16.9 Build 150310 Rel.55295n	Lenovo IdeaPad Yoga 2, Arch Linux 4.4.1	Success
WPA PSK	`12345678`	test-wpa-psk	TP-LINK, TL-WR841N, 3.16.9 Build 150310 Rel.55295n	HTC Desire 500, Android 4.1.2	Success
WPA PSK	`12345678`	test-wpa-psk	TP-LINK, TL-WR841N, 3.16.9 Build 150310 Rel.55295n	Meizu m2 note, Android 5.1	Success
WPA PSK	`12345678`	test-wpa-psk	TP-LINK, TL-WR841N, 3.16.9 Build 150310 Rel.55295n	Lenovo IdeaPad Yoga 2, Arch Linux 4.4.1	Success
WPA2 PSK	`12345678`	test-wpa2-psk	TP-LINK, TL-WR841N, 3.16.9 Build 150310 Rel.55295n	HTC Desire 500, Android 4.1.2	Success
WPA2 PSK	`12345678`	test-wpa2-psk	TP-LINK, TL-WR841N, 3.16.9 Build 150310 Rel.55295n	Meizu m2 note, Android 5.1	Success
WPA2 PSK	`12345678`	test-wpa2-psk	TP-LINK, TL-WR841N, 3.16.9 Build 150310 Rel.55295n	Lenovo IdeaPad Yoga 2, Arch Linux 4.4.1	Success
WPA2 PSK	`CWGUJAJX`	UPC1234567	TP-LINK, TL-WR841N, 3.16.9 Build 150310 Rel.55295n	HTC Desire 500, Android 4.1.2	Success
WPA2 PSK	`CWGUJAJX`	UPC1234567	TP-LINK, TL-WR841N, 3.16.9 Build 150310 Rel.55295n	Lenovo IdeaPad Yoga 2, Arch Linux 4.4.1	Success