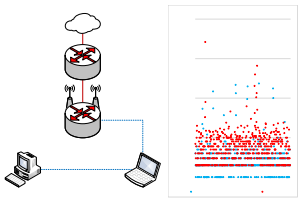
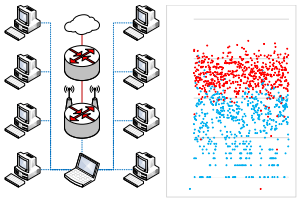
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2017
http://excel.fit.vutbr.cz

Automation of MitM Attack on WiFi Networks

Martin Vondráček*





Abstract
Widely used network technologies and principles of wireless security suffer weaknesses that can be exploited to perform the Man-in-the-Middle attack, allowing to eavesdrop or to spoof the network communication. The work focuses on possibilities of automation of the attack with a utilization of available specialized tools. The outcome of the research is the *wifimitm* package and the *wifimitmcli CLI* tool, both implemented in Python. The emphasis was placed on possibilities of further incorporation of the developed tool. The package provides 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 attacker.
This research can be used for automated penetration testing and forensic investigation. Finally, a 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.

Keywords: Man-in-the-Middle attack — accessing secured wireless networks — password cracking — dictionary personalization — tampering network topology — impersonation — phishing

Supplementary Material: Homepage — Demonstration Video — Slides — Downloadable Code

*xvondr20@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

This paper aims at research concerning a security of wireless networks. It delivers a study of widely used network technologies and principles of wireless security. Analyzed technologies and security algorithms suffer weaknesses that can be exploited to perform the Man-in-the-Middle attack. The *MitM* is a very dangerous network attack. A successful realization of this attack allows not only eavesdropping on all the victim's network traffic, but also spoofing the communication [1], [2, pp. 101–120], [3, pp. 4–7].

The aim of this research is to design, implement and test a tool able to automate the whole process of *MitM* attack on *WLAN*s. This way, using the automated tool would not require any action from the user. For the chosen implementation of the *MitM* attack, it is necessary for an attacker to be in the same *WLAN* as a victim. Therefore, this research focuses on exploitable weaknesses of particular 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. From the point when the attacker is connected to the network and topology is successfully modified, the attacker can start capturing or changing all the victim's network traffic.

Specialized tools focused on exploiting individual weaknesses in security algorithms currently used in *WLAN*s are already available. There are also specialized tools focused on individual steps of the *MitM*

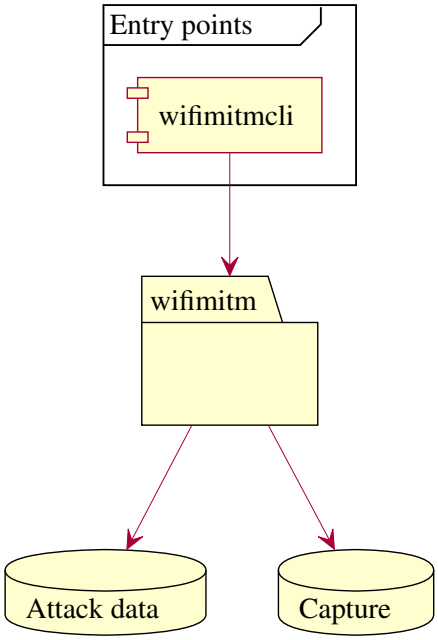


Figure 1. This figure shows the basic structure of the developed application. The tool *wifimitmcli* uses a functionality offered by the package *wifimitm*. The package is able to manipulate attack data useful for repeated attacks and capture files with intercepted traffic. Detailed structure of the package is described in section 3.

attack. Tools that were researched and selected for incorporation are outlined in section 2.

Based on the acquired knowledge, referenced researches and practical experience from manual experiments, the author was able to create an attack strategy. The plan is composed of an appropriate set of available tools. The strategy is then able to select and manage individual steps which are the most suitable for a successful *MitM* attack on given *WLAN*. This strategy further includes ways of impersonation and phishing for cases, when the network is properly secured and the weakest part of the security is the legitimate user.

The final solution was tested during experiments with an available set of equipment. Developed open source software, which could be incorporated into other tools, can perform a fully automated attack and requires zero knowledge. The use case of this software is found in automated penetration testing, forensic investigation, and education.

2. Security Weaknesses in WLAN Technologies

Following network technologies (Sections 2.1, 2.2), which find a significant utilization, unfortunately, suffer from security weaknesses in their protocols. These flaws can be used in the process of the *MitM* attack.

2.1 Wireless Security

Wired Equivalent Privacy (*WEP*) is a security algorithm introduced as a part of the IEEE 802.11 standard [4, p. 665], [5, pp. 1167–1169]. At this point, *WEP* is deprecated and superseded by subsequent algorithms, but is still sometimes used. *WEP* suffers from weaknesses and, therefore, it has been broken [6]. There are already implemented tools to provide access to wireless networks secured by *WEP* available [7]. Regarding *WEP* secured *WLAN*s, authentication can be either *Open System Authentication* (*OSA*) or *Shared Key Authentication* (*SKA*) [5, pp. 1170–1174]. In the case of *WEP OSA*, any *Station* (*STA*) can successfully authenticate to the *Access Point* (*AP*) [8, pp. 4–10]. *WEP SKA* provides authentication and security of transferred communication using a shared key. Confidentiality of transferred data is ensured by encryption using the *RC4* stream cipher. Methods used for cracking access to *WEP* secured networks are based on analysis of transferred data with corresponding *Initialization Vectors* (*IV*s).

Wi-Fi Protected Access® (*WPA*) was developed by the Wi-Fi Alliance® as a reaction to increasing number of security flaws in *WEP*. The main flaw of *WPA* security algorithm can be identified at the beginning of client device's communication, where an unsecured exchange of confidential information is performed during the four-way handshake. An attacker can obtain this unsecured communication and use it for consecutive cracking of the *Pre-Shared Key* (*PSK*).

Wi-Fi Protected Access® *2* (*WPA2*™) is a successor of *WPA*, but flaws of the *PSK* algorithm remain significant also for the *WPA2*. Information exposed during the handshake can be used for the dictionary attack, which can be improved by precomputing the *Pairwise Master Keys* (*PMKs*) [9, pp. 37–38], [10, p. 3].

A critical security flaw in wireless networks secured by *WPA* or *WPA2* is the functionality called *Wi-Fi Protected Setup*™ (*WPS*). This technology was introduced with an aim to provide a comfortable and secure way of connecting to the network. For a connection to the *WLAN* with *WPS* enabled, it is possible to use an individual *PIN*. However, the process of connecting to the properly secured network by providing *PIN* is very prone to brute-force attacks [11]. Because *WPS* is a usual 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].

[1] https://code.google.com/archive/p/

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*'s web administration. There are already implemented tools, which exploit relations between *SSID*s and default network passwords, e.g., *upc_keys*[2][3] by Peter Geissler. These tools could be used in an attack on the network with default *SSID* to improve dictionary attack using possible passwords.

2.2 Network Technologies Used in WLANs

DHCP is used to provide a network device with a suitable configuration without the need for intervention from the user [12]. *ARP* provides the mapping of *IP* address to the corresponding *MAC* address of the device in local area network based on *IPv4* [13]. *IPv6* utilizes features of *ICMPv6* to map devices in the local network with Neighbor Discovery. *IPv6* Neighbor Discovery provides similar functionality as *ARP* in *IPv4* networks.

Tampering network topology could be performed at the moment when an attacker is successfully connected to the target wireless network. *DHCP Spoofing* generates fake *DHCP* communication. This attack can also be referred to as *Rogue DHCP*. An attacker can perform this kind of attack to provide devices in the network with malicious configuration, most often a fake default gateway address or *DNS* address. A possible countermeasure, *DHCP Snooping*, is further described in the thesis [14]. A network attack called *ARP* Spoofing focuses on providing the network devices with fake *ARP* messages. 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 *MitM* position, he can persuade the victim about the mapping of default gateway's *IP* address to the attacker's *MAC* address. Possible defense against *ARP* Spoofing attack is analysis of *ARP* messages transmitted in the network – Dynamic ARP Inspection [14]. The absence of *ARP* in *IPv6* does not guarantee immunity to the attacks based on a very similar principle as *ARP* Spoofing. An attacker can send a specially generated *ICMPv6* 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 *MitM* attack. Possible defense is a Neighbor Discovery Inspection, as described in the thesis [14].

reaver-wps/

[2] https://haxx.in/upc-wifi/

[3] UPC company is a major ISP in Czechia, URL: https://www.upc.cz

2.3 Available Tools for Specific Phases of the MitM Attack on Wireless Networks

From perspective of the intended functionality of the implemented tool, the whole process of *MitM* attack on wireless networks can be divided into three main phases: *Accessing wireless network*, *Tampering network topology* and *Capturing network traffic*, as explained in Figure 2.

To access secured wireless networks, *Aircrack-ng suite*[4] is considered a reliable software solution. Considering the phase *Accessing wireless network* (Figure 2), following tools were 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*, *airodump-ng* and *upc_keys* can be utilized for cracking *WEP OSA*, *WEP SKA*, *WPA PSK* and *WPA2 PSK*. The tool *wifiphisher*[5] can be used to perform impersonation and phishing. Connection to the wireless network can be established by *netctl*[6]. *MITMf*[7] with its *Spoof* plugin can be used during the *Tampering network topology* phase. *Capturing traffic* can be done by the tool *dumpcap*[8], which is part of the *Wireshark*[9] distribution. Behaviour, usage and success rate of individual tools, as well as possibilities of controlling them by the implemented tool, were analyzed. The software selected for individual tasks of the automated *MitM* attack were chosen from the researched variety of available tools based on performed manual experiments, as described in the thesis [14].

3. Attack Automation Using Developed wifimitm Package and wifimitmcli Tool

The implemented tool is currently intended to run on *Arch Linux*[10], but it could be used on other platforms which would satisfy specified dependencies. This distribution was selected because it is very flexible and lightweight. Python 3.5 was selected as a primary implementation language for the automated tool and Bash was chosen for supporting tasks, e.g., installation of dependencies on *Arch Linux* and software wrappers.

The functionality implemented in the *wifimitm* package could be directly incorporated into other software products based on Python language. This way

[4] http://www.aircrack-ng.org/

[5] https://github.com/sophron/wifiphisher

[6] https://www.archlinux.org/packages/core/any/netctl/

[7] https://github.com/byt3bl33d3r/MITMf

[8] https://www.wireshark.org/docs/man-pages/dumpcap.html

[9] https://www.wireshark.org/

[10] https://www.archlinux.org/

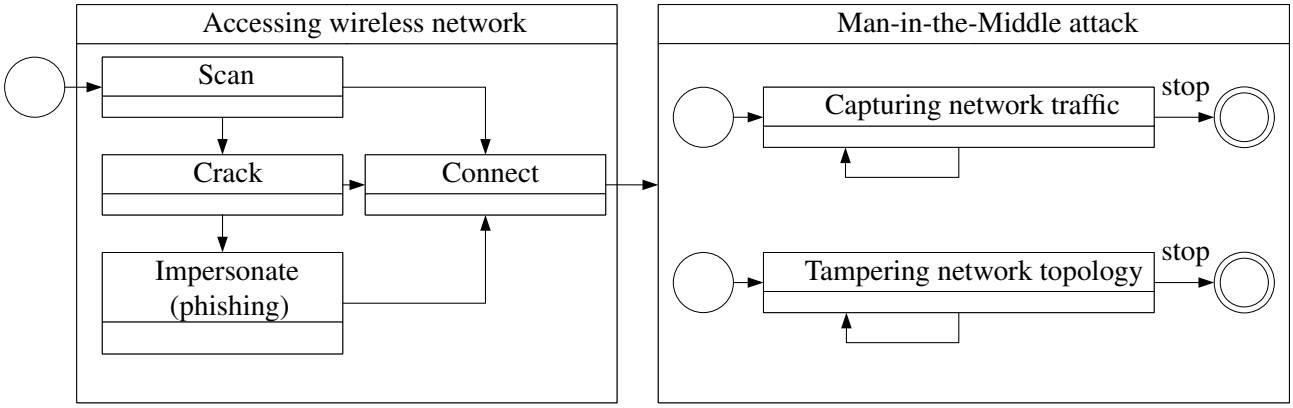


Figure 2. During the first phase – *Accessing wireless network*, the tool is capable of an attack on *WEP OSA*, *WEP SKA*, *WPA PSK* and *WPA2 PSK* secured *WLAN*s. In a case of the dictionary attack on the device deployed by the UPC company, used dictionaries are personalized by the implicit passwords. In the case of properly secured *WLAN*, impersonation (phishing) can be employed. Using this method, an attacker impersonates the legitimate network to obtain the *WLAN* credentials from the user. For the second phase – *Tampering network topology*, *ARP Spoofing* technique was selected from the researched methods. This method proved itself with reasonable performance during experiments. During this phase, the tool needs to continuously work on keeping the network *STA*s persuaded that the spoofed topology is the correct one. An attacker is now able to capture or modify the traffic. The successful *MitM* attack is established.

the package would work as a software library. Schema of the *wifimitm* package is in Figure 1.

The *wifimitm* package consists of following modules. The `access` module offers an automated process of cracking selected *WLAN*. It uses modules `wep` and `wpa2`, which implement attacks and cracking based on the used security algorithm. The `wep` module is capable of fake authentication with the *AP*, *ARP replay* attack (to speed up gathering of *IV*s) and cracking the key based on *IV*s. In the case of *WPA2* secured network, the `wpa2` module can perform a dictionary attack, personalize used dictionary and verify a password obtained by phishing. Verification of the password and dictionary attacks are done with a 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 *STA*s from selected *AP*.

If a dictionary attack against a correctly secured network fails, the `impersonation` module can manage a phishing attack. 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.

3.1 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 obtained in order to start an attacking procedure. Fake authentication in *WEP SKA* secured network needs *PRGA XOR* obtained from a detected authentication. Dictionary attack against *WPA PSK* and *WPA2 PSK* secured networks requires a captured handshake. Finally, for the 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. Data from successful attacks could be even shared between users of the implemented tool.

3.2 Dictionary Personalization

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*[11] 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.

3.3 Requirements

The implemented automated tool depends on several other tools, which are being controlled. The Python package can be automatically installed by its setup including Python dependencies. Non-Python dependen-

[11] https://haxx.in/upc-wifi/

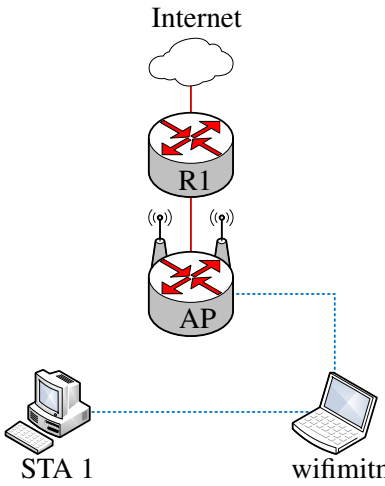


Figure 3. This figure shows the network topology used for success rate measurements and the first performance testing. Results of this performance testing are in Figure 4.

cies can be satisfied by installation scripts and wrappers, which are currently developed for *Arch Linux*.

MITMf has a number of dependencies. Therefore, the installation script also creates a virtual environment dedicated to *MITMf*. After installation, *MITMf* can be easily run encapsulated in its environment. *Wifiphisher* is also installed in its own environment and run using a wrapper. Tool *upc_keys* is compiled during installation. Some changes in *wifiphisher*'s source code were implemented, the installation script therefore applies a software patch. Other software dependencies are installed using a package manager.

Due to the nature of concrete 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 forged packets, the wireless network interface also needs to be capable of packet injection. 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. Managing monitor mode of interface is possible with the *airmon-ng* tool.

4. Experiments Concerning Various Devices and Attack's Performance Impact

The testing *WLAN* consisted of 1 *STA* and 1 *AP* connected to the Internet. A scheme of the networks used for the experiments is shown in Figures 3 and 5. The *STA* was correctly connected to the *AP* and it was successfully communicating with the Internet. The implemented *wifimitmcli* tool was then started and

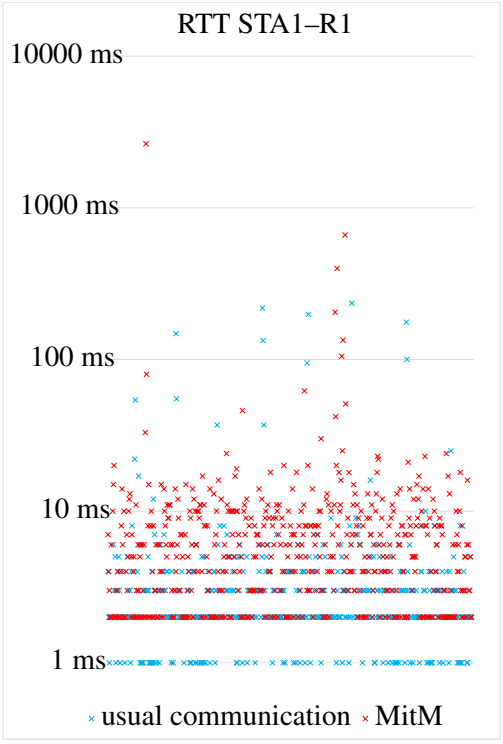


Figure 4. The first *WLAN* for performance testing was the same as for the success rate measurements described in section 4. Figure shows comparison of the measured *RTT* between *STA1* and *R1* during usual communication and during successful *MitM* attack. The results show the performance impact is not critical. Discussion with the users of the attacked network proved this attack unrecognizable.

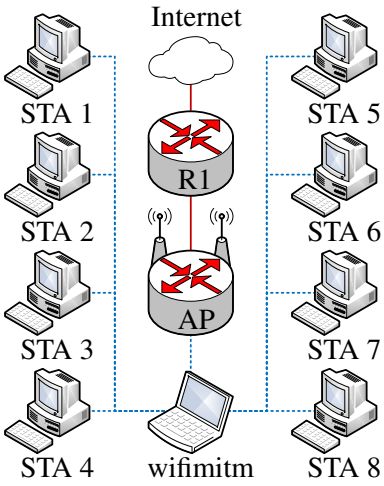


Figure 5. This figure shows the network topology consisting of 8 *STA*s and 1 *AP* which was used for the second performance testing. Results of this performance testing are in Figure 6.

automatically attacked 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*.

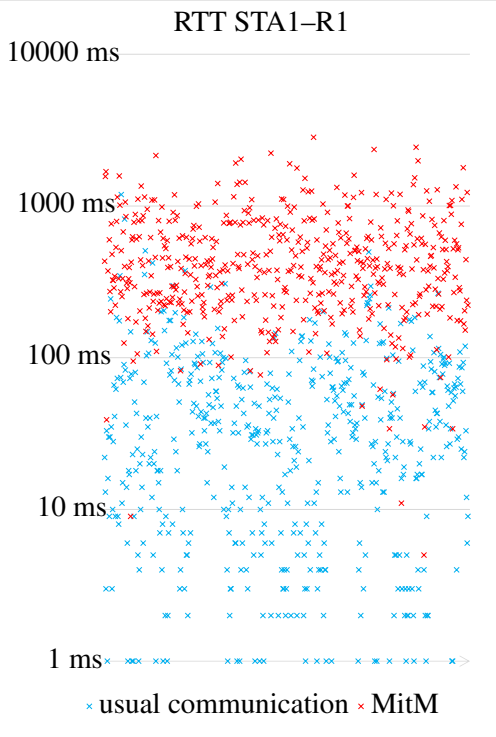


Figure 6. The second *WLAN* for performance testing consisted of 8 *STA*s and 1 *AP* connected to the Internet – streaming videos, downloading large files and browsing the web. The figure compares the *RTT* between *STA1* and *R1* during usual communication and during successful *MitM* attack. As we can see, the performance impact is more severe than in Figure 4. Despite the performance impact, the users of the network had no suspicion that they are being attacked.

Results of experiments [14] show, that open networks can be very easily attacked. *WEP OSA* and *WEP SKA* secured networks can be successfully attacked even if they use a long random password. *WPA PSK* and *WPA2 PSK* secured networks suffer from weak passwords (dictionary attack), default passwords and mistakes of users.

The performance impact of the *wifimitm* was compared using two *WLAN* setups based on SOHO[12] environment. Both experiments were also evaluated based on the fact, whether the attack being performed was revealed or whether the users had any suspicion about the malicious transformation of their *WLAN*. As Figures 4, 6 show, *MitM* attack using the *wifimitm* is successfully feasible in the target environments.

5. Conclusions

The goal of this research was to implement a tool that would be able to automate all the necessary steps to perform the *MitM* attack on *WLAN*s. The author searched for and analyzed a range of software and methods focused on penetration testing, communication sniffing and spoofing, password cracking and hacking in general. To be able to design, implement and test the tool capable of such attack, knowledge of different widespread security approaches was essential. The author further focused on possibilities of the *MitM* attack even in the case that given *WLAN* is secured correctly. Therefore, methods and tools for impersonation and phishing were researched and analyzed.

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 *WLAN*s. The developed package is, therefore, beneficial for others because 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 *CLI*. The implemented software comes with a range of additions for a convenient usage, e.g., requirements installation scripts for *Arch Linux*, requirements check, a Python package setup with *wifimitmcli* installation and even 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 perform an automated *MitM* attack on *WLAN*s successfully.

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 [15]. A software capable of automated *MitM* attack on *WLAN*s 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.

In the future iterations of the development, the product could focus on exploiting the weaknesses of widely used *WPS*. Concerning the current state of the product, it does not focus on enterprise *WLAN*s, which suffer their weaknesses as well.

This research was published as a bachelor's thesis and a software product in *NES@FIT* research group in May 2016, the author received dean's award and rector's award later that year.

[12] small office/home office

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 sincerely appreciate their help and professional advice.

References

[1] F. Callegati, W. Cerroni, and M. Ramilli. Man-in-the-middle attack to the HTTPS protocol. *Security Privacy, IEEE*, pages 78–81, Jan 2009.

[2] Stacy Prowell, Rob Kraus, and Mike Borkin. Chapter 6 - man-in-the-middle. In Stacy Prowell, Rob Kraus, and Mike Borkin, editors, *Seven Deadliest Network Attacks*, pages 101–120. Syngress, Boston, 2010.

[3] Martin Vondráček. Secure tunnel using Diffie-Hellman key agreement protocol, Feige-Fiat-Shamir identification protocol, AES and SHA256, 2017. http://www.stud.fit.vutbr.cz/~xvondr20/KRY/Secure-tunnel.pdf.

[4] F. Halsall. *Computer Networking and the Internet*. Addison-Wesley, 2005.

[5] 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. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pages 1–2793, March 2012.

[6] Scott Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the key scheduling algorithm of RC4. In Serge Vaudenay and AmrM. Youssef, editors, *Selected Areas in Cryptography*, Lecture Notes in Computer Science, pages 1–24. Springer Berlin Heidelberg, 2001.

[7] Erik Tews, Ralf-Philipp Weinmann, and Andrei Pyshkin. Breaking 104 bit WEP in less than 60 seconds. In Sehun Kim, Moti Yung, and Hyung-Woo Lee, editors, *Information Security Applications*, Lecture Notes in Computer Science, pages 188–202. Springer Berlin Heidelberg, 2007.

[8] Pieter Robyns. Wireless network privacy. Master's thesis, Hasselt University, Hasselt, 2014.

[9] Vishal Kumkar, Akhil Tiwari, Pawan Tiwari, Ashish Gupta, and Seema Shrawne. Vulnerabilities of wireless security protocols (WEP and WPA2). *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 1(2):34–38, 2012.

[10] 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)*, pages 1–4, Sept 2010.

[11] Craig Heffner. Cracking WPA in 10 hours or less – /dev/ttys0, 2011. http://www.devttys0.com/2011/12/cracking-wpa-in-10-hours-or-less/.

[12] R. Droms. Dynamic Host Configuration Protocol. Technical Report 2131, Internet Engineering Task Force, March 1997.

[13] D. Plummer. Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. Technical Report 826, Internet Engineering Task Force, November 1982.

[14] Martin Vondráček. Automatizace útoku MitM na WiFi sítích. Bakalářská práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2016.

[15] Jan Pluskal, Petr Matoušek, Ondřej Ryšavý, Martin Kmeť, Vladimír Veselý, Filip Karpíšek, and Martin Vymlátil. Netfox detective: A tool for advanced network forensics analysis. In *Proceedings of Security and Protection of Information (SPI) 2015*, pages 147–163. Brno University of Defence, 2015.