MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Practical man-in-the-middle attacks in computer networks

BACHELOR'S THESIS

Matěj Plch

Brno, Spring 2015

Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Matěj Plch

Advisor: RNDr. Jiří Kůr, Ph.D.

Acknowledgement

I wish to express my sincere thanks to RNDr. Jiří Kůr, Ph.D. for his guidance and valuable advices during work on this thesis.

Abstract

Using public wireless networks pose a real threat to the privacy of users due to the possibility of man-in-the-middle-attack. This thesis documents the issue of wireless access points operated by an attacker. Methods of creating an access point and executing attacks against its users are described in detail, including example usage of software tools and possible countermeasures against the attacks. In practical part the success and impact of attack tools is evaluated.

Keywords

man-in-the-middle, wireless networks, fake access point, sslstrip, SSLsplit, certificate, Karma

Contents

1 **Introduction** 1
2 **Man-in-the-middle attacks** 3
2.1 ARP cache poisoning 3
2.2 DNS man-in-the-middle attack 4
2.3 Man-in-the-middle attack using rogue DHCP server 5
2.4 SSL/TLS man-in-the-middle attack 6
2.5 Fake access point 7
3 **Fake access point** 8
3.1 Setting up wireless access point 8
3.1.1 Required hardware 8
3.1.2 Required software 9
3.2 Types of the access points 10
3.3 Creating evil twin access point 11
3.3.1 Cloning specific access point 12
3.3.2 Karma attack 12
3.4 Luring victims 13
3.4.1 Access point denial 14
3.4.2 Client luring 15
3.5 Countermeasures 15
4 **SSL/TLS man-in-the-middle attacks** 17
4.1 HTTPS stripping attack 17
4.1.1 Sslstrip 17
4.1.2 Countermeasures 20
4.2 SSL/TLS certificate forgery 21
4.2.1 SSLsplit 23
4.2.2 Mitmproxy 25
4.2.3 Countermeasures 26

5 **Practical tests** 29
5.1 Creating access point 29
5.2 Sslstrip 31
5.3 SSLsplit 33
5.4 Karma attack 36
6 **Conclusions** 38

1 Introduction

Due to the recent increase in popularity of wireless networks, people became inured to using them for everyday communication. Users now expect availability of wireless networks in restaurants, hotels, schools, and other public places, and do not hesitate to use them for accessing sensitive information, for example, bank accounts or corporate databases. Such a behaviour pose a threat to the privacy of users, because if an access point (AP) is under control of someone with nefarious intentions, then this AP can be used for monitoring of and tampering with communicated data. This type of attack, when an attacker controls communication from the position between a client and a server, is called man-in-the-middle (MITM) attack.

The aim of this thesis is to explore possibilities of MITM attacks in computer networks, with focus on attack techniques using fake AP. We describe principles of practical MITM attacks, and also demonstrate practical usage of tools implementing these attacks. The tools are then used to evaluate security of selected applications against MITM attacks. Our goal is also to recommend best practices how to mitigate presented attacks.

The structure of this thesis is following. First, we describe the principle of MITM attack and give examples of this attack in computer networks. In Chapter 3, we summarize what hardware and software requirements have to be met for creating an AP capable of executing attacks against connected users. Then we describe how to clone an AP to create an evil twin AP, even automatically by using the Karma attack, and give recommendations how to avoid connecting to such evil AP. Chapter 4 deals with practical MITM attacks compromising encrypted communication. We describe how these attacks work and show usage of software

tools for executing them. For each of these attacks we also describe countermeasures how to mitigate the possibility of being a victim. In the Chapter 5 we present results of our practical testing of described attacks utilizing fake AP. In the last chapter we conclude how are the described attacks successful.

2 Man-in-the-middle attacks

In this chapter, we describe what an MITM attack is. We describe principles of this attack and give examples of MITM attacks that work on different layers of computer network.

Basic idea of an MITM attack is that an attacker interposes himself in between two communicating parties and starts controlling their communication. Suppose Alice wants to communicate with Bob. If a malicious entity, called Mallot, manages to insert himself into the path of the communication between Alice and Bob, then he becomes the MITM and gains abilities of the attacker in the Needham-Schroeder attack model [1], meaning he is capable of reading and changing communicated messages. From this moment on there exist in fact two communication channels, one between Alice and Mallot and the other one between Mallot and Bob. Every message sent between Alice and Bob goes through Mallot and can be manipulated without any side noticing.

2.1 ARP cache poisoning

ARP Cache Poisoning, or ARP Spoofing, is an attack working on the third layer of the ISO/OSI reference model [2]. The aim of this attack is to change routes of the traffic on a local network so the data goes through a malicious host. To achieve this behaviour a vulnerability in the Address Resolution Protocol (ARP) is exploited.

Purpose of the Address Resolution Protocol is to translate IP address of desired host to the corresponding MAC address on the local network. If one host wants to send data to a specific IP address, it broadcasts to the local network asking for MAC address associated with the given IP address. The host with requested IP address sends a unicast reply

with its MAC address. Flaw in this protocol is that there are no steps taken to verify whether the obtained pair of IP and MAC address is valid, giving the attacker a chance to forge responses claiming that somebody else's IP address belongs to his MAC address. Tricked host then stores this information to speed up future communications, which effectively redirects data to the attacker.

To obtain a position between the victim and the rest of the network an attacker must spoof MAC addresses of the victim and the gateway and accordingly forward the data between them.

Two variants of this attack exist depending of what type of control packet is manipulated. First option is to create forged ARP reply packet to poison cache of the victim with malicious translation of IP addresses to MAC addresses. However, most of the network devices ignore unsolicited replies. Second option is to forge ICMP packets. Internet Control Message Protocol (ICMP) offers various types of messages to obtain information about the network, such as *Echo Request*, *Echo Reply*, *Redirect* or *Time Exceeded*. The attacker sends the victim a specially crafted Echo Request message containing the attacker's MAC address and the IP address of, for example, the gateway as a source address, resulting in victim caching this malicious pair of addresses and sending data to the attacker instead of the gateway in the future.

2.2 DNS man-in-the-middle attack

Domain Name System (DNS) is a distributed system providing mapping between domain names and IP addresses. In the DNS protocol a client sends a request to the server with a domain name and the server responds with an IP address of the given domain. In the ISO/OSI model this protocol operates on the application layer, i.e. layer 7.

DNS MITM attack inserts an attacker between a victim and a DNS server, which enables him to provide incorrect information about mapping domain names to IP addresses [3]. The result is a victim wanting to visit regular websites but obtaining IP addresses of malicious sites belonging to the attacker.

One way how to reach the MITM position is to intercept DNS request from the victim and reply on them faster than the authentic DNS server, resulting in replies from the DNS server being dropped by the victim host because they are no longer needed. The attacker has a good chance to respond faster than the genuine server because of possible recursive resolution of the IP address in the hierarchical system of domain names, but to be sure to respond first the attacker can slow down the server by flooding it with a large amount of DNS requests.

2.3 Man-in-the-middle attack using rogue DHCP server

Dynamic Host Configuration Protocol (DHCP) is an application layer protocol used for distributing network configuration parameters on a local network. Such parameters are, for example, assigned IP address, address of a local DNS server or address of a default gateway.

If the attacker is capable of setting up his own DHCP server, then he can configure connected hosts in such a way that the address of a default gateway will be address of his machine, which will effectively re-route all the outgoing traffic to the attacker [4].

For the attack to be successful it is not sufficient to start broadcasting to the network that there is available a new DHCP server, because new hosts can still accept settings from the legitimate server. At first, services from all available DHCP servers must be denied, giving the attacker a space for controlling the local network. DHCP servers can be taken down

by either using the Denial of Service (DoS) attack or the address starvation attack. Principle of DoS attack is to flood server with requests to slow it down, the address starvation attack consists of continual claiming IP addresses from the DHCP server until it runs out of them, making the server not able to serve new hosts.

When a victim host accepts configuration from the malicious DHCP server, it then sends all data to the attacker instead of the legitimate gateway. Attacker accepts this data, possibly manipulates them and forwards modified packets to the gateway to cover the existence of the MITM. Disadvantage of this attack is that incoming traffic goes from the gateway directly to hosts, so the attacker sees and controls only outgoing traffic.

2.4 SSL/TLS man-in-the-middle attack

Secure Sockets Layer/Transport Layer Security (SSL/TLS) are cryptographic protocols operating on the presentation network layer, i.e. layer 6. Their goal is to establish a secure connection between two sides communicating over an insecure network. These protocols provide confidentiality and optionally also authenticity of the exchanged messages, however, an attacker can under given conditions run an MITM attack to bypass the encryption and gain access to the plaintext [5].

The attacker can force the victim not to use a secure connection at all by manipulating the victim's HTTP traffic to prevent him from redirecting to a secure HTTPS site.

Another way how to get access to the plaintext is to intercept certificate provided by the server and pass to the victim a forged one with public key of the attacker, tricking the victim into encrypting messages directly for the attacker.

MITM attacks on SSL/TLS encryption are described in more detail in the Chapter 4.

2.5 Fake access point

In order to obtain control over the traffic in the wireless network an attacker can set up his own malicious AP. Such AP would appear as a regular point for connecting to the network, but all the traffic going through will be observed and possibly manipulated by the attacker.

The attacker can either passively wait for connected hosts or actively lure them to his network by cloning existing APs or APs requested by the wireless devices.

This topic is covered in more detail in the Chapter 3.

3 Fake access point

Creation of a malicious AP is an effective, yet easy to deploy method of running an MITM attack. In the simplest form, the attacker creates a publicly accessible wireless network, waits for some users to connect, and monitors all their data traffic. The attacker can then under given conditions collect their personal information and credentials, for example, passwords, credit card numbers, etc. This type of attack can be potentially highly effective due to the popularity of Wi-Fi capable devices such as laptops, smartphones or tables and ubiquitous presence of wireless networks in restaurants, hotels, university campuses, office buildings and so on. Advanced methods of luring potential victims does also exist, capable of connecting users to the attackers network without their explicit permission.

In this chapter, we first describe what kind of hardware and software equipment is necessary for establishing a malicious AP. Next, we discuss how the AP can operate in situations when the connection is limited. At the end, we describe how to clone existing APs and how to disconnect users from the networks they are currently using.

3.1 Setting up wireless access point

3.1.1 Required hardware

Wireless AP can be created by using a wireless router or a mobile wireless device. Specialised network auditing tools in a form of wireless routers can be purchased, for example, WiFi Pineapple [6]. However, using a mobile wireless device as an AP, for example, laptop or smartphone, brings higher mobility and better performance. Modern mobile devices

are usually already equipped with tool for sharing wireless connection, this functionality is called tethering [7].

For better results an external antenna should be used to enhance the signal strength. Stronger signal makes the malicious AP visible to the devices in a wider range and increases a chance of the attack to be successful, because mobile devices try to connect to the known network with the best signal strength to maintain a stable connection. The most popular wireless adapters with external antennas that may serve as APs are Alfa AWUS036H, Alfa AWUS036NHR and TP-LINK TL-WN722N [8].

If we want to perform an analysis of traffic in a wireless network, the used wireless adapter should support switching to a monitor mode. In monitor mode we can see service messages broadcast in the network we listen to, for example, *Probe Request* messages important for the Karma attack described in the Section 3.3.2.

If connection to the internet is needed but there isn't available a connection to a standard computer network, a GSM module can be used to connect to the Internet through a mobile network.

3.1.2 Required software

Standard operating systems, such as Windows, Mac OS X, or Android, are equipped with a tethering functionality, but to be able to read or manipulate the data going through the fake AP, a special software is needed. In this section, we describe what tools are available for creating an AP and processing the data going through.

Access point creation

Although an option to create an AP is usually already available in standard operating systems, a more customizable way of doing so is often required. There are many tools for creating APs, providing options to, for example,

set Service Set Identifier (SSID, name of the network), MAC address, or operating channel. These tools can also integrate advanced attacks such as Karma or HTTPS stripping attack, which will be discussed later. The examples of such tools are *hostapd* [9], *airbase-ng* [10], *WebSploit Framework* [11], *Ghost Phisher* [12], *MANA Toolkit* [13], *Subterfuge* [14] or *easy-creds* [15].

Data manipulation

Attackers can choose from a variety of instruments to manipulate the communicated data in real time. The goal of such manipulation can be to, for example, infect victims with a malware or hack inside an encrypted communication to extend the MITM attack. The examples of such programs are *mitmproxy* [16], *sslstrip* [17], *SSLsplit* [18], *Ettercap* [19], *Mallory* [20] or *Metasploit* [21].

3.2 Types of the access points

Attacks using a fake AP can be divided by the type of the network they are connected to into three classes.

- *Offline AP* – malicious AP can be able to collect credentials even if it is not connected to the Internet. This attack uses methods based on a social engineering to trick the victims to think that they are really online. One method is to clone most popular web sites and provide these clones to the victims, who will enter their passwords and reveal them to the attacker. Another method is to set up a captive portal, a page where users will be redirected, and ask there for entering personal information or passwords. An example of this attack is to ask for a password to an existing wireless network.

- *Rogue AP* – rogue AP is a wireless AP installed in a private network, but not authorized and managed by the legitimate network administrators [22]. Such an AP can be created by an employee, using tethering on his mobile device, to have a personal AP. Another option is that the AP is set up by an attacker to perform MITM attacks on users accessing resources in the private network, for instance, servers or printers. Both types of a rogue AP can possibly provide an unauthorized access to the private network and spy on users of the rogue AP.

- *Regular AP* – such AP provide a full access to the Internet and try to collect as many data as possible: passwords, visited websites, or emails. It is possible to use tools for targeting encrypted communication, such as sslstrip [17] or SSLsplit [18], to reject an encrypted connection or to forge encryption keys and be able to decrypt the data.

3.3 Creating evil twin access point

Evil twin is a malicious clone of an existing AP, that tries to convince unsuspecting victims to connect to the attacker's network. We describe methods how to clone an existing AP and how to persuade users and their mobile devices to connect to this evil AP. An attacker can choose to clone a specific AP, or to use an advanced technique called the Karma attack to listen which networks the wireless devices request and create them on demand.

3.3.1 Cloning specific access point

To successfully create a clone of an AP, the attacker must copy the SSID and use the same authentication method. Such an AP is, for the wireless devices, almost indistinguishable from the original one, and they can under given circumstances automatically connect to it. Limitation of this method is that the attacker is not able to successfully clone a protected wireless network without knowledge of an appropriate keys.

3.3.2 Karma attack

The attack technique called Karma exploits a vulnerability in the method of discovering available Wi-Fi APs [23]. When a mobile device wants to connect to a wireless network, at first it must perform a scanning for available networks and choose one with respect to its list of preferred networks. As described in the IEEE 802.11 standard [24], which specifies the Wi-Fi networks, there are two types of scanning, passive and active. The passive scanning consists of listening for *Beacon* management frames, which are continuously broadcast by APs. The active scanning involves sending *Probe Request* frames with SSIDs of preferred networks and listening for *Probe Response* frames from APs with the requested SSID. SSID contained in a Probe Request can be set to a name of a specific network, or left blank, which means that every AP in range should send a response. Sending Probe Requests with specified SSID is required for connecting to the networks with hidden SSID, because other methods of scanning will not discover their names. AP to connect to is then chosen with respect to the signal strength, authentication method, recent usage, or by other parameters.

The vulnerability is the fact that an active scanning for a specific networks reveals the list of preferred networks to the attacker. He can then

respond by sending appropriate Probe Responses and effectively pretend that he is the requested AP. The message flow of the Karma attack is shown in the figure 3.1.

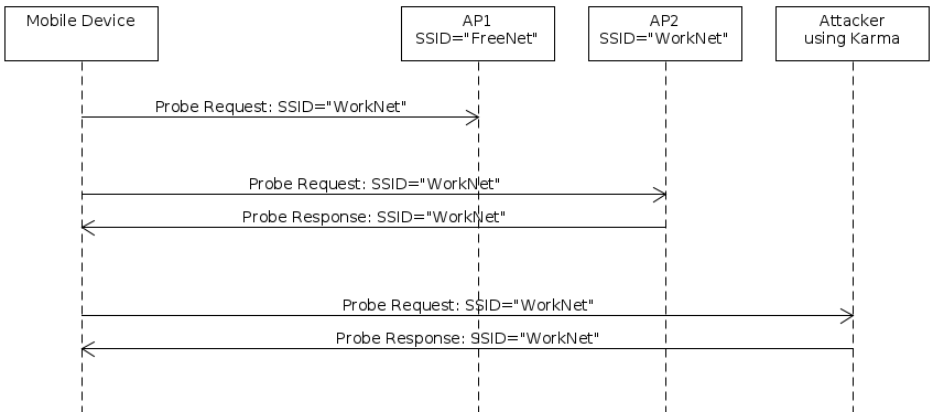


Figure 3.1: Karma attack

The Karma attack is named after the original set of tools exploiting this vulnerability [25], but it is also available in other tools, e.g., in the Metasploit Framework as *Karmetasploit* [26] or in a form of patches for hostapd [27], a daemon for creating software APs.

3.4 Luring victims

When the evil twin AP is up and running, it is time to force the victims to disconnect from the current network and lure them to the malicious one. Following methods describe how to ensure that the victim will disconnect from the network which he currently uses and how to increase a chance that he will connect to the malicious one.

3.4.1 Access point denial

- *Deauthentication* – the IEEE 802.11 standard specifies a special *deauthentication/disassociation* control frames, whose purpose is to tell the target wireless device that it is no longer authenticated/associated. Reception of such frame results in an immediate loss of connection. These control messages are not requests, but notifications, and devices have to obey them. By continuous broadcasting of these frames the attacker can deny the victims the connection to a legitimate AP, the only special step he needs to take is to spoof the source MAC address as if the frame has been sent from the legitimate AP. Even networks protected by encryption are vulnerable because most control frames are not encrypted. An amendment IEEE 802.11w [28] has been proposed to protect all control messages by encryption, excluding the ones needed for association. However, this has been proved to be just a half measure [29], because a flood of such frames induces a large amount of cryptographic computations, which prevent the unauthorized disconnections, but slows down the victim significantly.

- *Traffic flooding* – another way of making it impossible to connect to an AP is to flood the AP with a large amount traffic, making it not able to operate the network. Unlike the previous method, this attack cannot be targeted on an individual victim, because it will affect all the users connected to the AP.

- *Radio frequency jamming* – the most aggressive method of disconnecting a victim from a wireless network is to jam a radio channel with noise, which effectively brings down all the networks set up on the affected channel. The evil twin AP then must operate at least 5

channels away to avoid the jamming, because Wi-Fi radio channels are partially overlapping.

3.4.2 Client luring

Wireless devices prefer to connect to APs with the best signal strength to maintain a stable connection. By emitting stronger signal we increase our chances of convincing victims to connect to our AP. This can be done with a special antenna. For some wireless network adapters, such as those mentioned in the Section 3.1.1, it is also possible to manually increase the output power. This action can be illegal in some countries, for example, in the Czech Republic the upper limit for Wi-Fi signal strength is 20 dBm [30], nonetheless some wireless adapters are capable of operating with higher output power.

Following commands demonstrate how to increase signal strength on supported devices:

```
# set regulatory domain to Bolivia to allow 30 dBm
iw reg set BO
iwconfig wlan1 txpower 30
```

3.5 Countermeasures

Threats presented in this chapter result from an insecure usage of wireless networks by users or from a software programmed in an insecure way. It is possible to mitigate these risks using following methods:

- *Do not connect to unsecured networks* – use only well known networks protected by strong authentication mechanisms, e.g. WPA-PSK or WPA-802.1X. Also make sure the networks do not appear on places where they should not exist.

- *Clear list of used networks* – this measure will prevent device from possible broadcast of the names of used networks, effectively mitigating the Karma attack.

- *Turn Wi-Fi off when not used* – prevents device from connecting to a wireless network in a place where it should not exist.

- *Regularly update software* – Karma exploits an insecure option from the IEEE 802.11 standard, implementations of the standard which do not probe for networks by their names are secure.

- *Use special protection applications* – there are applications for mobile operating systems whose purpose is to control behaviour of wireless connectivity to prevent the device from connecting to untrusted networks and broadcasting names of the used networks. Examples of such applications are *Wi-Fi Privacy Police* [31] or *Pry-Fi* [32] for Android.

4 SSL/TLS man-in-the-middle attacks

Attacker in the MITM position is able not only to read such communicated data, but also to change the data in real time. By data manipulation, it is under given circumstances, possible to circumvent encryption and read plaintext data.

In this chapter, we describe various attacks on SSL/TLS encryption. We cover in detail how these attacks work and also show tools that implement the attacks. We also discuss how to defend against these attacks in dedicated subsections. For every tool we provide brief example of its usage.

4.1 HTTPS stripping attack

HTTPS is a standard protocol for secure communication over the network. It is an HTTP protocol encapsulated within an SSL/TLS encryption.

It is common that when people try to connect to a website where they have to log in securely, they do not specify which protocol should be used, e.g., they type in ”example.com” instead of ”https://example.com”. But the browser still ends up connecting to the website over HTTPS due to the server response containing *Location* header, redirecting the browser to the secure version of the website. Another situation, when browsers initiate HTTPS connection, is when a user clicks on a link with the HTTPS protocol already specified.

4.1.1 Sslstrip

Sslstrip by Moxie Marlinspike [17] implements the HTTPS stripping attack. In this attack, the internet browser of a victim is prevented from upgrading HTTP connections to the secure HTTPS protocol.

When sslstrip is run, it starts stripping from the traffic all the control data which would result in upgrading HTTP connections to the HTTPS. At first, it starts to remove HTTPS addresses from websites by looking for an HTML hyperlink tags "<a href='https://...” and replacing them with "<a href='http://...", so the affected users will not click on an HTTPS hyperlink. Next thing sslstrip does is removing Location headers from HTTP responses, so the browsers won't get redirected to an HTTPS version of the website.

But web servers often require secured connection and does not allow clients to proceed using only HTTP. This is why sslstrip intercepts the connection between the client and the server and establishes two separate connections, one with the client using HTTP and another one with the server using HTTPS. Now the affected client sends his data unencrypted, but the server is unaware of the attack. To make this work sslstrip maintains a map of stripped addresses to correctly translate between HTTP and HTTPS connections.

Additionally, sslstrip has to deal with following problems:

- *Data compression* – present compression of websites is an obstruction for parsing contents of websites to alter HTTPS hyperlinks to HTTP, solution is to add an HTTP headers preventing using of compression.

- *Sessions* – if the browser uses for logging in stored session cookie instead of username and password, then the password does not get compromised. The session cookie can be used for impersonating the victim, but the session can eventually expire. On the other hand, knowledge of the victim's password is much more powerful. It will provide access to other victim's accounts due to common password reuse. To be able to record the log in procedure, sslstrip redirects

requests with session cookies to the same address, but with headers Set-Cookie that expires all the cookies in the request. Such killing of sessions can be suspicious, so sslstrip can be configured to wait, for example, 5 minutes, and after this time to start killing only newly encountered sessions, so it will not cause all the sessions being terminated at once.

- *Secure cookies* – internet browsers will refuse to transmit secure cookies over unencrypted HTTP, so sslstrip removes from the headers information that the cookies are secure.

Example usage:

```
# route TCP traffic to the port used by sslstrip
iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j
    REDIRECT --to-port 10000
# start sslstrip
# -f show padlock favicon in the browser to make the websites seem
    secured
# -k kill newly encountered sessions to force the victim to
    re-enter passwords
# -l 10000 set listen port
sslstrip -f -k -l 10000
```

Sslstrip will now intercept all HTTP connections and will tamper the traffic as described above. It will also add a padlock icon, which will be shown in the browser, to make it look like a secured connection is used. All the sessions encountered by sslstrip, which won't be present in its database of active sessions, will be terminated, and the client will have to send again his password, but now over an unencrypted connection. Every intercepted HTTP message will be stored in the log file sslstrip.log.

4.1.2 Countermeasures

- *Request encrypted connection* – when accessing websites requiring entering sensitive information, always specify used protocol by preceding the website address with "https://".

- *Check whether secure protocol is used* – internet browsers usually provide positive feedback by showing green address bar or a lock icon in the address bar area for connections established over HTTPS.

- *Virtual Private Network (VPN)* – principle of VPN is establishing of an encrypted connection with a remote private network over an unprotected public network, usually the Internet. This ensures that the traffic is protected against reading and manipulation on its way over the network.

- *Use browser extensions enforcing HTTPS* – there are available extensions offering to automatically enable HTTPS when visiting websites. They use predefined or user-defined lists of websites to determine for which sites HTTPS should be used. One example is an extension *HTTP Nowhere* [33], which enforces secured connection for all the websites. As a result, all websites using only HTTP are inaccessible, and user has to explicitly allow individual unsecured websites which he wants to access. Example of a less strict extension is *HTTPS Everywhere* [34], which has a predefined list of sites where to require HTTPS. This extension does not block HTTP-only sites and user can extend the list of HTTPS websites. However, sites not contained in the list are still vulnerable to the HTTPS stripping.

- *HTTP Strict Transport Security (HSTS)* – HSTS [35] is an Internet standard allowing sites to enforce using encrypted connection. On the first visit of a website, client receives response header called *Strict-Transport-Security* containing information about using strict transport security, which will be stored and the browser will always require secure connection for the website. Unfortunately the client is vulnerable on the first visit of a website, because sslstrip removes HSTS headers. Furthermore, two tools trying to outsmart HSTS has been developed. One is called *SSLStrip+* [36] and uses a custom DNS server to change domain names in such a way that the browser will not have HSTS associated with the new domain. Another tool, called *Delorean* [37], uses Network Time Protocol (NTP) to tell the victim a false time to invalidate HSTS records. Both tools have limited practical application because SSLStrip+ is an experimental proof of concept implementation and Delorean provides false time only when the victim updates its clock.

4.2 SSL/TLS certificate forgery

Usual client-server communication secured by SSL/TLS proceeds as follows:

1. Client initiates communication with the server.

2. Server responds with a certificate containing domain name, public key, and signature.

3. Client validates the certificate. Performed checks are for example if the certificate is signed by a trusted certification authority (CA), if it is issued for the correct domain name, or if it is not expired.

4. If the given certificate passes the validation, a secure client-server communication is established, using server's public key.

Figure 4.1 shows how this scheme can be compromised using MITM attack. The attacker splits the communication into two independent connections, one between himself and the client, and the other one between himself and the server. Then he receives a certificate from the server, replaces the contained public key with his own one, and sends this forged certificate to the client. If the client accepts the forged certificate, then a connection between the client and the attacker is established. The attacker then establishes connection between himself and the server and resends data between the client and the server to make the connection seem intact, but is able to read and manipulate the exchanged data.

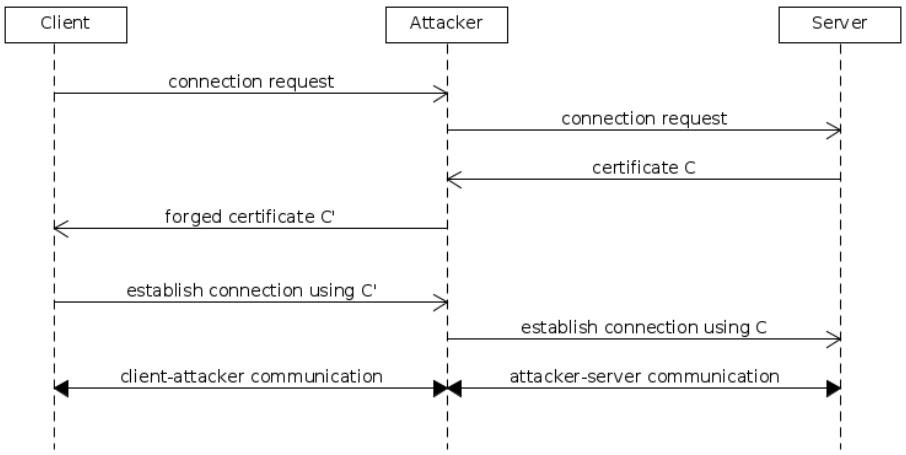


Figure 4.1: SSL certificate forgery

For this attack to be successful, the victim client has to accept provided forged certificate. If the client's software, for example, an internet browser or an email client, does not validate the certificates properly, then a forged one will be accepted without any notification. Modern internet browsers

validate certificates of web pages and show a warning when a certificate is not signed by a trusted CA. The victim then usually has an option to add an exception and use the untrusted certificate, or leave the page. However, according to surveys [38], most users simply ignore certificate warnings and just click through them to continue surfing. A way how to avoid showing SSL warnings is to install own CA certificate into the victims computer, this can be done by offering the certificate on a captive portal or by manual installation when the attacker has a physical access to the victim's computer.

4.2.1 SSLsplit

SSLsplit [18] is a tool for performing MITM attacks against SSL/TLS communication. It terminates SSL/TLS connections and initiates new connections to the original destination address, then it forwards and logs all the transmitted data. For SSL/TLS connections it generates and signs X509v3 certificates on-the-fly. Supported protocols for interception are plain TCP, plain SSL/TLS, HTTP, and HTTPS.

Additional features:

- Removes response headers of *Public Key Pinning Extension for HTTP* (HPKP), whose purpose is to use a particular certificate for the website, allowing SSLsplit to pass to the victim a forged one.

- Denies *Online Certificate Status Protocol* (OCSP) requests to prevent victims from checking revocation status of provided certificates.

- Prevents switching to the experimental protocols *QUIC/SPDY* by Google, which aim to provide better speed and security.

- High performance – SSLsplit is developed with performance in mind, its multithreaded, event based architecture tries to maximize amount of connections it is able to handle.

Example usage [39]:

First, we need to create a certificate of our CA:

```
# generates RSA key pair
openssl genrsa -out ca.key 4096
# create certification authority certificate
openssl req -new -x509 -days 3560 -key ca.key -out ca.crt
```

Next, we run SSLsplit using following commands:

```
# enable packet forwarding in the operating system
sysctl -w net.ipv4.ip_forward=1

# setup iptables for prerouting desired protocols to the SSLsplit
iptables -t nat -F # flush nat table
# prerouting rules for different protocols
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT
    --to-ports 8080 # (HTTP)
iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT
    --to-ports 8443 # (HTTPS)
iptables -t nat -A PREROUTING -p tcp --dport 587 -j REDIRECT
    --to-ports 8443 # (STARTTLS SMTP)
iptables -t nat -A PREROUTING -p tcp --dport 465 -j REDIRECT
    --to-ports 8443 # (SSL SMTP)
iptables -t nat -A PREROUTING -p tcp --dport 993 -j REDIRECT
    --to-ports 8443 # (SSL IMAP)
iptables -t nat -A PREROUTING -p tcp --dport 5222 -j REDIRECT
    --to-ports 8080 # (messaging connections)

# SSLsplit requires existing folders for logging:
mkdir /tmp/sslsplit
mkdir /tmp/sslsplit/logdir
```

```
# Start SSLsplit:
# -D - debug mode, run in foreground and show debug messages
# -l connections log - logfile for connections summary
# -j /tmp/sslsplit - change root directory for log files
# -S logdir/ - folder for connection logs
# ssl 0.0.0.0 8443 - proxy specification (connection type, listen
    address, port)
# tcp 0.0.0.0 8080 - similar as with ssl
# -k ca.key - private key of our certification authority
# -c ca.crt - certificate of our certification authority
sslsplit -D -l connections.log -j /tmp/sslsplit/ -S logdir/ ssl
    0.0.0.0 8443 tcp 0.0.0.0 8080 -k ca.key -c ca.crt
```

4.2.2 Mitmproxy

Mitmproxy [16] is an interactive MITM proxy for HTTP, capable of on-the-fly SSL certificate generation. It provides console interface for visualization of intercepted HTTP requests and responses (see Figure 4.2), with an ability to delay messages for manual modification. For an automated traffic analysis and modification, a library *libmproxy* is present for writing Python scripts performing data manipulation.

Mitmproxy can operate in various modes of operation:

- *Regular proxy* – clients have to be configured to use mitmproxy as a regular proxy.
- *Transparent mode* – mitmproxy is intercepting traffic and forwards it to the desired destination, useful when it is not possible to configure the client's proxy settings, for example, when running on an AP.
- *Reverse proxy* – mitmproxy acts as a regular HTTP server.

- *Upstream proxy* – unconditionally transfers all requests to another defined proxy.

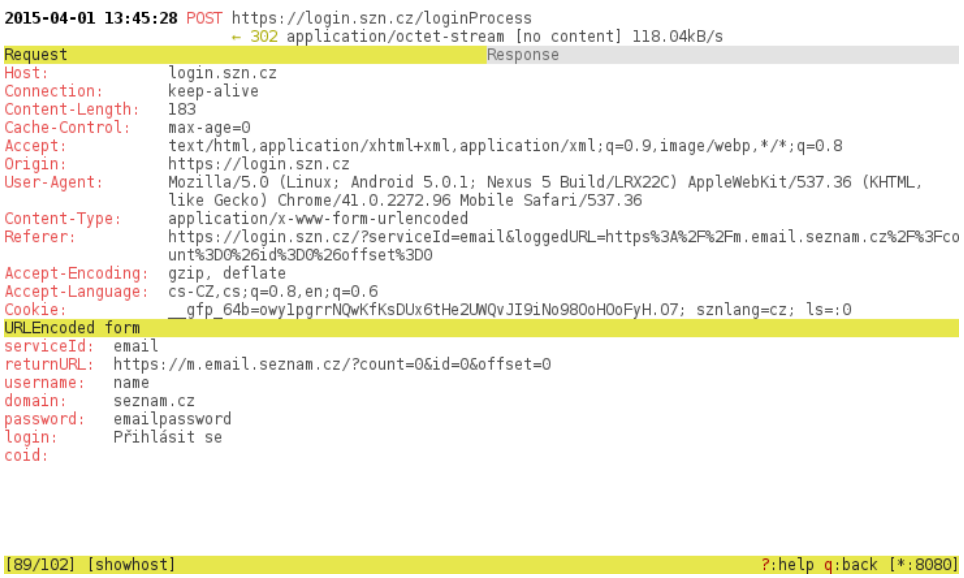
Example usage [40]:

```
# enable packet forwarding
sysctl –w net.ipv4.ip_forward=1
# redirect HTTP and HTTPS to mitmproxy
iptables –t nat –A PREROUTING –i wlan1 –p tcp ––dport 80 –j
    REDIRECT ––to–port 8080
iptables –t nat –A PREROUTING –i wlan1 –p tcp ––dport 443 –j
    REDIRECT ––to–port 8080
# start mitmproxy
# –T – use transparent mode
# ––host – use value in the Host header for URL display
mitmproxy –T ––host
```

4.2.3 Countermeasures

- *Do not confirm certificate exceptions* – untrusted certificates should only be accepted under specific circumstances, doing so during regular internet surfing is highly dangerous.
- *Manually check fingerprints of certificates* – this option requires obtaining original certificate from a trusted source and comparing it with the one used by an internet browser, however, this method is too complicated for everyday usage.
- *HSTS* – this does not only defend against HTTPS stripping, as described in the section 4.1.2, it does also mitigate SSL certificate forgery by forbidding usage of untrusted certificates.
- *Certificate pinning* – very efficient way of securing SSL communication is to have the public key already installed in the program [41],

(a) Intercepted traffic in mitmproxy



(b) HTTP POST message containing password

Figure 4.2: Mitmproxy user interface

because if the key is already present, the program doesn't need to rely on public key infrastructure and trust certificates obtained over the Internet.

- VPN – this measure protects data traffic as described in the section 4.1.2, but user has to take special care during authentication of the VPN server. If the server authenticates using an SSL/TLS certificate, this certificate also has to be verified.

5 Practical tests

In this chapter, we present results of our testing of the tools presented in previous chapters. We describe how we set up an AP and used it to perform selected MITM attacks against victims in a laboratory environment. We also discuss success rate of each attack and resulting behaviour of victims.

5.1 Creating access point

For creating an AP capable of performing MITM attacks, we used an ordinary laptop with installed *Kali Linux* [42]. Kali Linux is a penetration testing Linux distribution, offering a collection of pre-installed testing software. Another reason for choosing this distribution is a driver support for chosen wireless adapters, which enables them to operate in a monitor mode. We used TP-LINK TL-WN722N as a wireless adapter with sensitive external antenna.

Software used for creating an AP was hostapd. A DHCP server was created using *dnsmasq*. Configuration has been done similarly to the evil AP recipe [43] provided by Offensive Security, developers of Kali Linux.

AP configuration script:

```
#!/bin/bash
NET=wlan0 # the Internet
WLAN=wlan1 # our wireless network

## setup dnsmasq
sed -i 's#^DAEMON_CONF=.*#DAEMON_CONF=/etc/hostapd/hostapd.conf#'
    /etc/init.d/hostapd

cat <<EOF > /etc/dnsmasq.conf
```

```
log-facility=/var/log/dnsmasq.log
interface=$WLAN
dhcp-range=10.0.0.10,10.0.0.250,12h
dhcp-option=3,10.0.0.1
dhcp-option=6,10.0.0.1
log-queries
EOF

# start DHCP server
/usr/sbin/service dnsmasq start

## setup hostapd
/sbin/ifconfig $WLAN up
/sbin/ifconfig $WLAN 10.0.0.1/24

# re-route traffic from the wireless network to the Internet
/sbin/iptables -t nat -F
/sbin/iptables -F
/sbin/iptables -t nat -A POSTROUTING -o $NET -j MASQUERADE
/sbin/iptables -A FORWARD -i $WLAN -o $NET -j ACCEPT
echo '1' > /proc/sys/net/ipv4/ip_forward

cat <<EOF > /etc/hostapd/hostapd.conf
interface=$WLAN
driver=nl80211
ssid=Free
channel=1
EOF

# start access point
/usr/sbin/service hostapd start
```

5.2 Sslstrip

For executing the HTTPS stripping attack, we used a pre-installed sslstrip in the Kali Linux distribution. For testing how successful this attack can be, we have chosen 5 world's most visited websites [44] and 5 most visited Czech websites [45], according to Alexa[1]. To the tests we have also included internet banking of the 3 biggest Czech banks [46]. We have then accessed them from the most used web browsers [47] Google Chrome 40, Internet Explorer 11 (IE) and Mozilla Firefox 36, and operating systems Windows 8.1 and Kubuntu 14.04. In the test, we tried to log in and examined if the attacker was able to steal user credentials. For testing purposes, we have cleaned cache of used browsers to prevent them from using information about HTTPS usage from previous sessions.

Results of our tests are presented in the table 5.1. Symbol ✓ denotes successful defence of the browser or website against sslstrip, when secure HTTPS connection was used, whereas symbol ✗ means that the attack was successful, insecure HTTP was used and we were able to eavesdrop user's credentials. The results indicate that there is no difference among used operating systems, results on Windows 8.1 and Linux distribution Kubuntu 14.04 were identical. However, there were differences in behaviour of the tested browsers. Google Chrome appeared to be the most secure, thanks to its predefined list of websites using HSTS and requiring connection over HTTPS. But the list contains only a very few popular sites, almost all of the other sites were still vulnerable to this attack. Firefox performed with a similar security level, but its predefined list of websites using HTTPS is noticeably shorter. Windows-only browser Internet Explorer failed in this test, because current version 11 does not support HSTS.

1. Alexa is commonly cited company providing analytical data about web traffic.

	Machine A			Machine B	
	Chrome	IE	Firefox	Chrome	Firefox
google.com	✓	✗	✓	✓	✓
facebook.com	✓	✗	✗	✓	✗
yahoo.com	✓	✗	✓	✓	✓
baidu.com	✗	✗	✗	✗	✗
amazon.com	✗	✗	✗	✗	✗
seznam.cz	✗	✗	✗	✗	✗
heureka.cz	✗	✗	✗	✗	✗
centrum.cz	✗	✗	✗	✗	✗
aukro.cz	✗	✗	✗	✗	✗
alza.cz	✗	✗	✗	✗	✗
ib24.csob.cz	✗	✗	✗	✗	✗
servis24.cz	✗	✗	✗	✗	✗
kb.cz	✓	✓	✓	✓	✓

Legend:
✓ - successfully defends against sslstrip
✗ - vulnerable against sslstrip
Machine A - Windows 8.1, Lenovo Y50-70
Machine B - Kubuntu 14.04, Lenovo G700

Table 5.1: Results of testing sslstrip

The only website resistant to this attack on its own is an internet banking of *Komerční banka*. Users of their internet banking authenticate using personal certificates, and the website refuse to provide the login form over an insecure connection. Further investigation of how this security mechanism works is out of scope of this thesis and is left for future work.

We were also not able to intercept password in plaintext for the internet banking of *Česká spořitelna*, but we have managed to intercept the client's response. Further inspection of the JavaScript component calculating response out of the client's password is out of scope of this thesis, but it is not believed to be secure.

5.3 SSLsplit

We have used SSLsplit already installed in the Kali Linux, and tested it with the same websites and browsers as we tested with the sslstrip in the previous section. The aim of our tests was to see if browsers correctly show warnings for websites providing invalid certificates and if the browsers block users from accessing such websites which are also present on the predefined HSTS lists.

Results presented in the table 5.2 show similar results as when we tested sslstrip, but we have also managed to reveal one serious security flaw. Symbol ✓ marks situations where user was prevented from entering website using an untrusted certificate, symbol ⚠ denotes that a browser warned the user about the untrusted certificate, but he was still able to continue to that site. The most insecure situation represents symbol ✗, where a browser treated the untrusted certificate as trusted.

On the machine B, running on Kubuntu 14.04, browsers always gave the user at least a warning about invalid certificate, and blocked access

	Machine A			Machine B			Machine C	
	Chrome	IE	Firefox	Chrome	IE	Firefox	Chrome	Firefox
google.com	✗	✗	✗	✓	⚠	✓	✓	✓
facebook.com	✗	✗	✗	✓	⚠	⚠	✓	⚠
yahoo.com	✗	✗	✗	✓	⚠	✓	✓	✓
baidu.com	✗	✗	✗	⚠	⚠	⚠	⚠	⚠
amazon.com	✗	✗	✗	⚠	⚠	⚠	⚠	⚠
seznam.cz	✗	✗	✗	⚠	⚠	⚠	⚠	⚠
heureka.cz	✗	✗	✗	⚠	⚠	⚠	⚠	⚠
centrum.cz	✗	✗	✗	⚠	⚠	⚠	⚠	⚠
aukro.cz	✗	✗	✗	⚠	⚠	⚠	⚠	⚠
alza.cz	✗	✗	✗	⚠	⚠	⚠	⚠	⚠
ib24.csob.cz	✗	✗	✗	⚠	⚠	⚠	⚠	⚠
servis24.cz	✗	✗	✗	⚠	⚠	⚠	⚠	⚠
kb.cz	✗	✗	✗	✓	✓	✓	✓	✓

Legend:
✓ - user prevented from entering credentials
⚠ - showed SSL/TLS warning, user is able to continue
✗ - forged certificate treated as trusted
Machine A - Windows 8.1, Lenovo Y50-70
Machine B - Windows 8.1, Lenovo G700
Machine C - Kubuntu 14.04, Lenovo G700

Table 5.2: Results of testing SSLsplit

to the websites protected by HSTS. But on the machine A with Windows 8.1 no certificate warnings or errors were shown, so the attack was 100% successful. We have repeated the tests on a different machine with Windows 8.1 (Machine C), but this time browsers correctly warned the user as on Kubuntu.

Examination of this problem on the affected machine, Lenovo Y50-70, revealed that all the SSL certificates provided by the websites were issued by the same CA, called *Superfish* [48]. But Superfish is not a trusted CA, it is an American company developing visual search tools. Their software intercepted all the HTTP communications, even those secured by SSL/TLS, and injected web pages with targeted advertisements. Superfish was able to manipulate SSL/TLS secured connections because their certificate was installed as a trusted root CA, giving it capability to forge certificates of any website. It is usual that a specialized software, for example, an antivirus program, intercepts SSL/TLS connections, but it has to properly validate presented certificates in the same way as, for example, web browser. However, Superfish does not take necessary steps to validate certificates, and masks potential forged certificates. As a result, internet browsers on the affected machine never received certificates obtained from the Internet, but only certificates modified by Superfish. Browsers treated such certificates as trusted and Superfish gained access to all the communication protected by HTTPS.

Superfish is not a malware, it is an adware pre-installed in the system by the manufacturer Lenovo. Existence of this adware capable of SSL/TLS interception, which we independently discovered, was reported to the public on 19^{th} February 2015 [49], even global media informed about this incident, for example, BBC [50]. These articles inform about Superfish performing MITM attack on HTTPS connections, but they do not mention that Superfish fails to validate certificates and allows

MITM attacker to forge certificates without any notice, as we discovered. However, this vulnerability has been independently discovered by other researchers [51].

5.4 Karma attack

In this subsection, we describe practical tests of the Karma attack.

Our goal was to find a device vulnerable to the Karma attack, test how well does the attack work and try to mitigate the vulnerability.

To test available devices for this vulnerability, we used programs *airmon-ng* and *Wireshark*. We enabled monitor mode on a supported wireless adapter using a command *"airmon-ng start wlan1"* and examined raw wireless network traffic in Wireshark for Probe Request frames with names of requested networks.

The vulnerable device found was smartphone Samsung Galaxy S II with operating system CyanogenMod 11, based on Android 4.4.4. This device was constantly probing for one preferred network, and if it couldn't connect for approximately 90 seconds, it started probing for 16 previously used networks, protected or unprotected.

We have tested the Karma attack using a tool called *MANA Toolkit* [13], available in the Kali Linux repositories. This software performs standard Karma attack, but also adds extra features to make the attack more successful. The most significant enhancement is a so-called loud mode, in which the program builds a list of all the networks requested by network devices, and presents the whole list of networks to all the devices. Another improvement is that MANA does not try to clone a protected network, because it is not effectively possible without knowledge of the password, but instead it creates an unprotected network with the same name to trick the user to connect manually.

MANA was able to clone networks probed for by the Samsung phone, which have subsequently successfully connected to the cloned network and we were able to intercept its communication.

To mitigate this vulnerability, it was sufficient to install an application called Wi-Fi Privacy Police, which takes control over network probing and prevents from sending names of preferred networks.

We have also tried to estimate how frequent are such vulnerable devices. We have listened for a Wi-Fi traffic using monitor mode for an hour in an office building, and examined obtained data for unnecessary network probing. We have identified 658 different wireless devices by their MAC addresses, from which approximately 108 were only APs. Inspection of Probe Requests broadcast by the mobile devices revealed 31 potentially vulnerable devices, which accounts for approximately 5.6 % of the 550 present wireless devices. These devices were sending a large number of Probe Requests with names of networks, from which a significant amount were unsecured public networks. Such behaviour not only makes the devices vulnerable to the Karma attack, but also violates privacy of users. Revealed SSIDs of networks can potentially be used for monitoring which locations users visit. By the MAC addresses we identified manufacturers of the devices, which were Huawei, Sony, Microsoft, Xiaomi, HTC, Apple, Intel, Samsung, LG, Lenovo, and ASUS. However, results of this measurement are very approximate, because individual wireless devices behave differently, and once connected, they may not probe for other networks. Considering that majority of the devices in the office building are usually associated with some work network, the real percentage of vulnerable devices may be much higher than 5.6 %.

6 Conclusions

This thesis reviewed the possibilities of MITM attacks in computer networks, with focus on attacks using a fake AP. First we have described what is a principle of MITM attack and what types of this attack exist in computer networks.

In the following chapters we studied possibilities of using fake AP for MITM attacks. We described what are the requirements for creating a fake AP and how to convince victims to connect to it. Next, we described what attacks can the attacker use to compromise an SSL/TLS protected communication, with examples of concrete software tools and their usage. We also describe how to defend against the presented attacks.

Final chapter is dedicated to the results of our practical tests. We showed that the HTTPS stripping attack is still feasible in practice, in spite of the fact that it is not a recent vulnerability. Testing of the attack using forged SSL/TLS certificates confirmed improvements in the security of internet browsers, thanks to using the HSTS technology. And finally, we showed that the Karma attack is still a real threat for some wireless devices.

Our tests also revealed a severe security flaw present in the recent Lenovo laptops. We have discovered that an adware Superfish intercepted HTTPS communications by issuing its own trusted certificates, and this behaviour potentially masked ongoing MITM attack using forged certificates. This vulnerability has been independently discovered and published by other researchers, so we didn't have to file a report to the manufacturer.

Bibliography

[1] R. M. Needham and M. D. Schroeder, “Using Encryption for Authentication in Large Networks of Computers”, *Commun. ACM*, vol. 21, no. 12, pp. 993–999, Dec. 1978.

[2] W. Enck, “ARP Spoofing”, English, in *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg and S. Jajodia, Eds., Springer US, 2011, pp. 48–49, ISBN: 978-1-4419-5905-8.

[3] A. Singh, B. Singh, and H. Joseph, “Vulnerability Analysis for DNS and DHCP”, English, in *Vulnerability Analysis and Defense for the Internet*, ser. Advances in Information Security, A. Singh, B. Singh, and H. Joseph, Eds., vol. 37, Springer US, 2008, pp. 111–124, ISBN: 978-0-387-74389-9.

[4] A. Wong and A. Yeung, “Address Configuration and Naming”, in *Network Infrastructure Security*, Springer US, 2009, pp. 137–179, ISBN: 978-1-4419-0165-1.

[5] G. Nath Nayak and S. G. Samaddar, “Different flavours of Man-In-The-Middle attack, consequences and feasible solutions”, in *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, vol. 5, Jul. 2010, pp. 491–495.

[6] HAK5. (2015). WiFi Pineapple, [Online]. Available: `https://www.wifipineapple.com/` (visited on 05/10/2015).

[7] Wikipedia contributors. (2015). Tethering, [Online]. Available: `http://en.wikipedia.org/wiki/Tethering` (visited on 05/10/2015).

[8] Raymond.CC Blog. (2013). Best Compatible USB Wireless Adapter for BackTrack 5, Kali Linux and Aircrack-ng, [Online]. Available: `https://www.raymond.cc/blog/best-compatible-usb-wirele`

ss-adapter-for-backtrack-5-and-aircrack-ng/ (visited on 05/10/2015).

[9] J. Malinen. (2013). hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator, [Online]. Available: https://w1.fi/hostapd/ (visited on 05/10/2015).

[10] T. d'Otreppe. (2010). Airbase-ng, [Online]. Available: http://www.aircrack-ng.org/doku.php?id=airbase-ng (visited on 05/10/2015).

[11] F. Allahverdinazhand. (2014). WebSploit Advanced MITM Framework, [Online]. Available: http://sourceforge.net/projects/websploit/ (visited on 05/10/2015).

[12] S. E. Ekiko. (2015). Ghost Phisher, [Online]. Available: https://github.com/savio-code/ghost-phisher (visited on 05/10/2015).

[13] Sensepost. (2015). Improvements in Rogue AP attacks – MANA, [Online]. Available: https://www.sensepost.com/blog/2015/improvements-in-rogue-ap-attacks-mana-1-2/ (visited on 05/10/2015).

[14] M. Toussain and C. Shields. (2013). Subterfuge, [Online]. Available: https://code.google.com/p/subterfuge/ (visited on 05/10/2015).

[15] E. Milam. (2013). Easy-creds, [Online]. Available: http://easy-creds.sourceforge.net/ (visited on 05/10/2015).

[16] A. Cortesi. (2015). Mitmproxy, [Online]. Available: https://mitmproxy.org/ (visited on 05/10/2015).

[17] M. Marlinspike. (2011). Sslstrip, [Online]. Available: http://thoughtcrime.org/software/sslstrip/ (visited on 05/10/2015).

[18] D. Roethlisberger. (2015). SSLsplit – transparent and scalable SSL/TLS interception, [Online]. Available: https://www.roe.ch/SSLsplit (visited on 05/10/2015).

[19] A. Ornaghi, M. Valleri, E. Escobar, and E. Milam. (2015). Ettercap, [Online]. Available: http://ettercap.github.io/ettercap/index.html (visited on 05/10/2015).

[20] Intrepidus Group. (2013). Mallory: Transparent TCP and UDP Proxy, [Online]. Available: https://intrepidusgroup.com/insight/mallory/ (visited on 05/10/2015).

[21] Rapid7. (2015). Metasploit, [Online]. Available: http://www.metasploit.com/ (visited on 05/10/2015).

[22] J. Geier. (2003). Identifying Rogue Access Points, [Online]. Available: http://www.wi-fiplanet.com/tutorials/article.php/1564431 (visited on 05/10/2015).

[23] D. Dai Zovi and S. Macaulay, "Attacking automatic wireless network selection", in *Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC*, Jun. 2005, pp. 365–372.

[24] "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)*, pp. 977–980, Mar. 2012.

[25] D. A. D. Zovi. (2006). KARMA Attacks Radioed Machines Automatically, [Online]. Available: http://theta44.org/karma/ (visited on 05/10/2015).

[26] Metasploit contributors, Offensive Security. (2014). Karmetasploit, [Online]. Available: http://www.offensive-security.com/metasploit-unleashed/Karmetasploit (visited on 05/10/2015).

[27] R. Wood. (2012). Karma, [Online]. Available: http://digi.ninja/karma/ (visited on 05/10/2015).

[28] "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 4: Protected Management Frames", *IEEE Std 802.11w-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, and IEEE Std 802.11y-2008)*, Sep. 2009.

[29] C. Liu and J. Yu, "Rogue Access Point Based DoS Attacks against 802.11 WLANs", in *Telecommunications, 2008. AICT '08. Fourth Advanced International Conference on*, Jun. 2008.

[30] The Czech Telecommunication Office, *General Authorisation No. VO-R/12/09.2010-12 for the use of radio frequencies and for the operation of equipment for wideband data transmission in the 2.4 GHz – 66 GHz bands*, General Authorisation, 2010. [Online]. Available: http://www.ctu.eu/164/download/VOR/VO-R_12_09-2010-12_en.pdf.

[31] Expertise Centre for Digital Media. (2015). Wi-Fi Privacy Police. Hasselt University, [Online]. Available: https://play.google.com/store/apps/details?id=be.uhasselt.privacypolice (visited on 05/10/2015).

[32] Chainfire. (2014). Pry-Fi, [Online]. Available: https://play.google.com/store/apps/details?id=eu.chainfire.pryfi (visited on 05/10/2015).

[33] C. Wilper. (2013). HTTP Nowhere, [Online]. Available: https://addons.mozilla.org/cs/firefox/addon/http-nowhere/ (visited on 05/10/2015).

[34] Electronic Frontier Foundation. (2015). HTTPS Everywhere, [Online]. Available: https://www.eff.org/https-everywhere (visited on 05/10/2015).

[35] OWASP. (2015). HTTP Strict Transport Security, [Online]. Available: https://www.owasp.org/index.php/HTTP_Strict_Transport_Security (visited on 05/10/2015).

[36] L. N. Egea. (2015). SSLStrip+, [Online]. Available: https://github.com/LeonardoNve/sslstrip2 (visited on 05/10/2015).

[37] J. Selvi, "Bypassing HTTP Strict Transport Security", 2014, Black Hat Europe.

[38] A. P. Felt, A. Ainslie, R. W. Reeder, S. Consolvo, S. Thyagaraja, A. Bettes, H. Harris, and J. Grimes, "Improving SSL Warnings: Comprehension and Adherence", 2015, Google; University of Pennsylvania. [Online]. Available: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43265.pdf.

[39] C. D. Nardo. (2014). ARP spoofing, MITM attack over SSL-splitted connections, and fake CA certificate forging. All together. Against an iPhone, [Online]. Available: http://www.claudiodinardo.com/arp-spoofing-mitm-attack-over-ssl-splitted-conne

ctions-and-fake-ca-certificate-forging-all-together-against-an-iphone/ (visited on 05/10/2015).

[40] P. C. Heckel. (2013). How To: Use mitmproxy to read and modify HTTPS traffic, [Online]. Available: http://blog.philippheckel.com/2013/07/01/how-to-use-mitmproxy-to-read-and-modify-https-traffic-of-your-phone/ (visited on 05/10/2015).

[41] OWASP. (2015). Certificate and Public Key Pinning, [Online]. Available: https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning (visited on 05/10/2015).

[42] Offensive Security. (2015). Kali Linux, [Online]. Available: https://kali.org (visited on 05/10/2015).

[43] Offensive Security. (2014). Kali Linux Evil Wireless Access Point, [Online]. Available: https://www.offensive-security.com/kali-linux/kali-linux-evil-wireless-access-point/ (visited on 05/10/2015).

[44] Alexa. (2015). The top 500 sites on the web, [Online]. Available: http://www.alexa.com/topsites (visited on 05/10/2015).

[45] Alexa. (2015). Top Sites in Czech Republic, [Online]. Available: http://www.alexa.com/topsites/countries/CZ (visited on 05/10/2015).

[46] Wikipedia contributors. (2015). Seznam bank působících v Česku. Czech, [Online]. Available: http://cs.wikipedia.org/wiki/Seznam_bank_p%C3%85%C2%AFsob%C3%83%C2%ADc%C3%83%C2%ADch_v_%C3%84%C2%8Cesku (visited on 05/10/2015).

[47] StatCounter. (2014). Top 5 Desktop Browsers on 2014, [Online]. Available: http://gs.statcounter.com/#desktop-browser-ww-yearly-2014-2014-bar (visited on 05/10/2015).

[48] Superfish. (2015). Superfish - Visual Search and Image Recognition, [Online]. Available: http://www.home.superfish.com/ (visited on 05/10/2015).

[49] Engadget. (2015). New Lenovo PCs shipped with factory-installed adware, [Online]. Available: http://www.engadget.com/2015/02/19/lenovo-superfish-adware-preinstalled (visited on 05/10/2015).

[50] BBC. (2015). Lenovo taken to task over 'malicious' adware, [Online]. Available: http://www.bbc.com/news/technology-31533028 (visited on 05/10/2015).

[51] F. Valsorda. (2015). Komodia/Superfish SSL Validation is broken, [Online]. Available: https://blog.filippo.io/komodia-superfish-ssl-validation-is-broken/ (visited on 05/12/2015).