

WIRELESS NETWORK SECURITY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ÇANKAYA UNIVERSITY

BY

ÇAĞLAR ÜLKÜDERNER

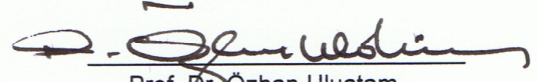
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2007

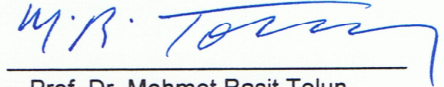
Title of the Thesis : **Wireless Network Security**

Submitted by : **Çağlar ÜLKÜDERNER**

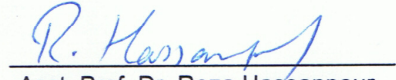
Approval of the Graduate School of Natural and Applied Sciences, Çankaya University


Prof. Dr. Ozhan Uluatam
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science/Computer Engineering.


Prof. Dr. Mehmet Raşit Tolun
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science/Computer Engineering.


Asst. Prof. Dr. Reza Hassanpour
Supervisor

Examination Date : 13.10.2007

Examining Committee Members (first name belongs to the chairperson of the jury and the second name belongs to supervisor)

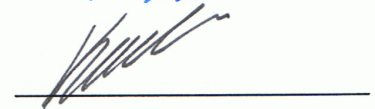
Asst. Prof. Dr. Murat Erten

(TOBB)




Dr. Abdülkadir Görür

(Çankaya Univ.)



Asst. Prof. Dr. Reza Hassanpour

(Çankaya Univ.)



STATEMENT OF NON PLAGIARISM

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Çağlar ÜLKÜDERNER

Signature :

Date

: 13.10.2007

ABSTRACT

WIRELESS NETWORK SECURITY

Ülküderner, Çağlar

M.S.c., Department of Computer Engineering

Supervisor : Asst Prof. Dr. Reza Hassanpour

SEPTEMBER 2007, 102 pages

This thesis includes a comparative study on wireless network security issues. Thesis also introduces a new method using a GSM operator which can control the internet accesses with improved security. By this method, mobile devices without GPRS cards can use internet access services using 802.11x connections and making acceptable investment by GSM operators.

Keywords: Wireless Network Security, One Time Password on Wireless Networks

ÖZ

KABLOSUZ AĞ GÜVENLİĞİ

Ülküderner, Çağlar

Yüksek Lisans, Bilgisayar Mühendisliği Anabilim Dalı

Tez Yöneticisi : Yar. Doç. Dr. Reza Hassanpour

Eylül 2007, 102 sayfa

Bu çalışma, kablosuz ağ güvenliğini incelemektedir. Çalışma, GSM operatörlerinin, internet erişim noktalarını nasıl kontrol edebileceği ve güvenliği arttırılmış yeni bir yöntem içermektedir. Bu yöntem ile mobil araçlar için GPRS kart kullanmadan, 802.11x bağlantısı kullanılarak, makul bir yatırımla GSM operatörleri üzerinden internet erişim hizmeti vermek mümkün olabilir.

Anahtar Kelimeler: Kablosuz Ağ Güvenliği, Kablosuz Ağlarda Tek Kullanımlık Şifre

ACKNOWLEDGMENTS

I would like to give my pleasures to my supervisor Asst. Prof. Dr. Reza Hassanpour for his great patience throughout my university life and thesis.

Also thanks for the great support to my brother Türker Gülüm.

I dedicate this thesis to my mom Gülay Ülküderner.

TABLE OF CONTENTS

STATEMENT OF NON PLAGIARISM	iii
ABSTRACT	iv
ÖZ	v
ACKNOWLEDGMENTS	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	x
LIST OF TABLES	xi
CHAPTERS:	
1 INTRODUCTION	1
1.1 Problem Definition and Motivation	2
1.2 Scope of Thesis	2
1.3. Main Challenges	3
1.4. Brief Introduction to Methods Used	3
2. FUNDAMENTALS OF WIRELESS NETWORK PROTOCOLS AND SECURITY ISSUES	4
2.1. IEEE 802.11 standards	5
2.2. Prevalence of Wireless and 802.11	7
2.3. General Security Problems on 802.11	8
2.3.1. EAP-MD5	8
2.3.2. EAP-OTP	10
2.3.3. EAP-GTC	10
2.4. Seven Problems on 802.11	10
2.4.1. PROBLEM #1: EASY ACCESS	10
2.4.2. Problem #2: "Rogue" Access Points	11

2.4.3. Problem #3: Unauthorized Use of Service	12
2.4.4. Problem #4: Service and Performance Constraints	13
2.4.5. Problem #5: MAC Spoofing and Session Hijacking.....	15
2.4.6. Problem #6: Traffic Analysis and Eavesdropping.....	16
2.4.7. Problem #7: Higher Level Attacks.....	17
3.Wi-Fi NETWORK TRAFFIC.....	19
3.1. Connection / Access Protocol in Wi-Fi Networks	19
3.2. Analyzing Wi-Fi Network Traffic	20
3.3. Information From All Frames.....	21
3.4. Information From Data Frames	22
3.5. Information From Management Frames.....	23
3.6. Summary of Wi-Fi traffic.....	24
4.WEP Overview	26
4.1. Decrypting data without keys	29
4.2. WEP IV problems.....	29
4.3. Some attacks	31
4.4. Problems with RC4	34
4.5. Cipher and mode of operation.....	35
4.6. Session key derivation	36
5.Wi-Fi PROTECTED ACCESS (WPA).....	38
5.1. Background information for WPA.....	38
5.1.1. WPA-PSK	38
5.1.2. Breaking Confidentiality	38
6.BREAKING THE SECURITY OF WI-FI.....	39
6.1. Recovering a Passphrase Seeded WPA Key.....	39
6.2. WI-FI Protected Access (WPA).....	41
6.3. Software Tools	42
6.3.1. KISMET.....	43
6.3.2. TCPDump	44
6.3.3. ETHREAL	45
6.3.4. Ettercap.....	46
6.3.5. IPTABLES.....	47

6.3.6. HOSTAP	47
6.3.7. WPA Supplicant	48
6.3.8. MadWiFi.....	48
7.GSM LOCATION BASED AUTHENTICATION USING SMS.....	49
7.1. Steps Of Authentication.....	50
8.CONCLUSION.....	52
REFERENCES	R1
APPENDICESY	A1
A. Acronyms & Abbreviations	A1
B. Used Configurations:.....	A6
B.1. hostapd.conf	A6
B.2. wpa_supplicant.conf.....	A3
B.3. wireless_ap configuration shell script.....	A3
B.4. SMS Server-Client Program.....	A4
B.4.1.corePortAccess.c	A4
B.4.2.messageSendingCore.c.....	A7
B.4.3.zaman.c	A19
B.4.4.sms_server.c.....	A20
B.4.5.sms_client.c	A26
B.4.6.Makefile.....	A29
B.4.7.SMSd	A30

LIST OF FIGURES

Figure 1 Simple Working Diagram.....	4
Figure 2 Eap-Md5 Choreography	9
Figure 3 The Protocols Of Connecting To A Wi-Fi Network.	19
Figure 4 Mac Frame Format.....	20
Figure 5 Frame Control Field.....	21
Figure 6 Capability Field Of The Beacon Frame.....	23
Figure 7 Kismet Screen Shot.....	43
Figure 8 Ethreal Screen Shot	45
Figure 10 Gsm Location Based Authentication Using Sms	50

LIST OF TABLES

Table 1 Information Available From An Analysis Of Wi-Fi Frames.....	24
---	----

CHAPTER 1

INTRODUCTION

There are many authentication techniques for wireless networks. Some of these techniques can be easily cracked by hackers yet others can be cracked by pattern matching, brute force attacks and similar techniques. Nowadays there are some additional authentication mechanisms such as radius servers for wireless authentication. They create user name password authentications for wireless access.

There are next generation security mechanisms which generate one time passwords. This mechanism uses time as input. After obtaining the time, it generates the password for predefined time space. By this method user's password is changed every time. User does not need to worry about his/her password.

This method is useful for public areas such as hotels, cafes, schools, offices etc. In such places every user will have a one time password generator and except them nobody can use the wireless network. This password generator can be configured by other services as well as wireless authentication server.

This method is not used for wireless authentication. This thesis is also related to an application using one time password generators with mobile phones in wireless network authentication

1.1 Problem Definition And Motivation

Mainly there are several authentication techniques on wireless access. The problem is that hacking those systems are not very complicated. The strongest one is WPA and it can be hacked in one hour on fair average. Most of the attackers use open source cracking program called **aircrack** [1]. We can use One Time Password (OTP) against these attacks. This can be achieved by using token cards (EAP-GTC) but this is not very useful. Nowadays we are using PDA, embedded computers etc. with or without USB ports or PCMCIA slots. This means that the token card is not meaningful for such devices in daily life. The alternative of this technology is OTP generators. The approach is using OTP with mobile operators. In public areas, the internet connection is becoming a problem. The security is the most important part of these connections. Mobile operators can be used for these operations. Short Message Services (SMS) can be used for password transactions. Also the requested area coordinates can be used as input.

Wireless systems are widely used in nowadays. Therefore, the main problem is restriction of users in wireless access. There is mobile communication almost everywhere. Mobile phones are a part of daily life. No one wants to remember passwords. On the other hand, no one wants anyone to get his or her passwords. For this reason, mobile technologies can be combined with wireless access authentication. This will be the simplest method for secure connection.

1.2 Scope Of The Thesis

This thesis includes a research on general wireless security mechanisms and their problems. It also introduces a new method of using securely one time password generators in public areas. Using this method, it is possible to merge mobile SMS technology, mobile location finder and wireless OTP technology.

1.3 Main Challenges

Our first goal is to analyze wireless security mechanisms and holes and offering different authentication methods.

There are two authentication mechanisms related to OTP in Extensible Authentication Mechanism (EAP). Those authentication mechanisms are named as EAP-OTP and EAP-GTC. EAP-OTP is based of EAP-MD5. Both use an external authentication server. This is also another problem.

The second goal is not to use any external authentication server with OTP. The authentication password can be changed by UNIX time but reentering the password is the main problem for an authenticated user. A session base system can be developed as a solution. After the authentication, the user will start the session and continue with a session period. At this point, session can be stolen by an attacker. However, first of all attacker must find the security code which is generated by the user using one time password generator. It can be resolved by pattern matching technique as the attacker can catch a specific pattern.

1.4 Brief Introduction To Methods Used

Open source architectures are used for testing and developing the OTP codes. First, a wireless access point (AP) which is a wireless card that can work on managed mode is constructed. For authentication, open source programs are used. A shell code which is just as the proof of concept for using one time password generator without EAP-OTP and EAP-GTC is developed. It has been started by analyzing the authentication mechanisms.

D-Link PCMCIA card with RA link chip set [2] for managed mode AP is used. For client, Intel based wireless card is used. As an open source

product **hostap**¹ driver for AP and **madwifi** [3] for authentication and **iptables** for routing the packets between the interfaces is chosen. As the operating system, **Fedora Core 5 Linux** Kernel 2.6.20-1.2312.fc5 is used. **Kismet** [4] for detecting access points, **Ettercap** for wireless sniffing, **Ethreal** and **Wire Shark** for sniffing and **TCPDUMP** for dumping the packets are used.

It is simply shown below (Figure 1):

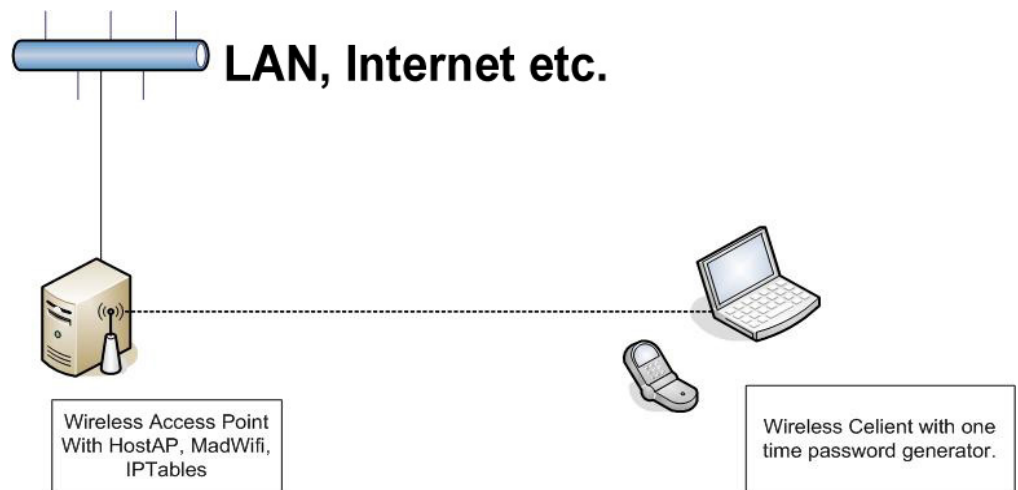


Figure 1 Simple Working Diagram

¹ Appendix 10.2.1 hostapd configuration, 10.2.2 wpa_supplicant configuration, 10.2.3 wireless_ap_configuration shell script

CHAPTER 2

FUNDAMENTALS OF WIRELESS NETWORK PROTOCOLS AND SECURITY ISSUES

2.1 IEEE 802.11 standards

These standards are also known as Wi-Fi. The purpose of these standards is to provide wireless connectivity to automatic machinery, equipment or stations that require rapid deployment, which may be portable or hand-held or which may be mounted on moving vehicles within a local area. These standards also offers regulatory bodies a means of standardizing access to one or more frequency bands for the purpose of local area communication.

The IEEE standards numbers and contents are shown below [5]:

IEEE 802.11, 1999 Edition (ISO/IEC 8802-11: 1999) IEEE Standards for Information Technology — Telecommunications and Information Exchange between Systems — Local and Metropolitan Area Network — Specific Requirements — Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications

IEEE 802.11a-1999 (8802-11:1999/Amd 1:2000(E)), 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 1: High-speed

Physical Layer in the 5 GHz band

IEEE 802.11b-1999 Supplement to 802.11-1999, Wireless LAN MAC and PHY specifications: Higher speed Physical Layer (PHY) extension in the 2.4 GHz band

802.11b-1999/Cor1-2001, 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 2: Higher-speed Physical Layer (PHY) extension in the 2.4 GHz band — Corrigendum1

IEEE 802.11d-2001, Amendment to IEEE 802.11-1999, (ISO/IEC 8802-11) 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: Specification for Operation in Additional Regulatory Domains

IEEE 802.11e-2005, 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 8: Medium Access Control (MAC) Quality of Service Enhancements

IEEE 802.11F-2003 IEEE Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation

IEEE 802.11g-2003 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: Further Higher-Speed Physical Layer Extension in the 2.4 GHz Band

IEEE 802.11h-2003 IEEE Standard for Information technology — Telecommunications and Information Exchange Between Systems — LAN/MAN Specific Requirements — Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Spectrum and Transmit Power Management Extensions in the 5GHz band in Europe

IEEE 802.11i-2004 Amendment to IEEE Std 802.11, 1999 Edition (Reaff 2003). IEEE Standard for Information technology — Telecommunications and information exchange between system — 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 Interpretation

IEEE 802.11j-2004 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 7: 4.9 GHz–5 GHz Operation in Japan

2.2 Prevalence of Wireless and 802.11

This chapter summarizes the currently available Local Area Wireless technologies that can be employed to build a wireless communication network. Wireless systems can be divided into three categories namely portable, fixed and IR (infrared) wireless systems.

- Portable wireless systems are battery-powered devices that can be used both inside and outside the wireless facility. Examples include laptops and personal digital assistants (PDAs).
- Fixed wireless systems require a line of sight for connection. They use fixed antennas, provide data connectivity as high as a cable and are used for high-speed data connection.
- IR wireless systems use infrared radiation to send signals. They have very short range. Examples are television remote controls, cordless computer keyboards and mouse.

Our thesis is based on the 802.11-based Wireless Local Area Networks (WLANs) Security, which comes under the category of portable wireless systems.

2.3 General Security Problems on 802.11

Many organizations are now considering deployment of wireless LANs and are working on the basic network designs before going to pilot projects as long as network security is concerned. The problems with security on 802.11 networks have been widely reported elsewhere. Network architects are now facing the challenge of designing secure networks in light of the known problems. This article will discuss seven of the most pressing wireless LAN security problems and potential designs that can mitigate the risk associated with each of them.

2.3.1 EAP-MD5

The EAP-MD5 is a Challenge Handshake Authentication Protocol (CHAP), as defined in RFC 1994. Figure 2 shows the choreography of the EAP-MD5 mechanism.

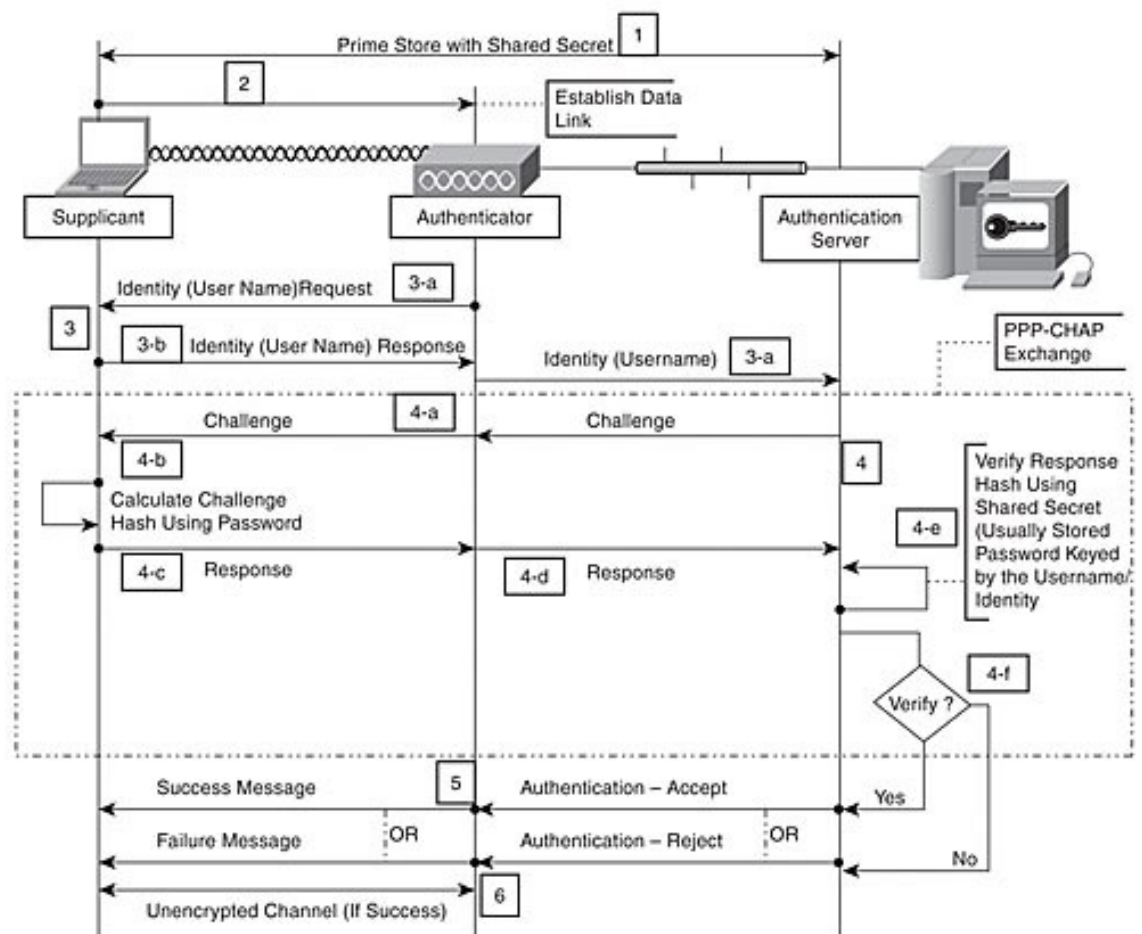


Figure 2 EAP-MD5 Choreography

For EAP-MD5 to work, the client and the authentication server must have a shared secret, usually a password associated with an identity/username. This needs to be established out of band (Step 1 in Figure 2). The connectivity (Step 2 in Figure 2) and identity exchange (Step 3 in Figure 2) are required before the EAP-MD5 process. The EAP-MD5 method consists of a random challenge to the supplicant (Step 4-a in Figure 2) and a response from the supplicant (Step 4-c, Step 4-d in Figure 1), which contains the hash of the challenge created using the shared secret (Step 4-b in Figure 2). The authentication server verifies the hash (Step 4-e in Figure 2) and accepts or rejects the authentication. The authenticator allows or disallows access (Step 5 in Figure 2) based on this decision. If successful, the supplicant gains access (Step 6 in Figure 2).

EAP-MD5 is a pure authentication protocol; after the authentication, the messages are transmitted in clear text. It is also a client authentication protocol - the server side (authenticator) is not authenticated; therefore, it cannot detect a rogue AP.

EAP-MD5 also contains a set of good features: It requires only lightweight processing (which translates to less hardware) and does not require a key/certificate infrastructure. Although pure EAP-MD5 has some value in the PPP world, it is of limited use in the wireless world. For example, Microsoft has dropped the support for EAP-MD5 for the wireless interface in Windows XP. Support was dropped because of security problems; EAP-MD5 is vulnerable to dictionary and brute-force attacks when used with Ethernet and wireless.

2.3.2 EAP-OTP

EAP-OTP is similar to MD5, except it uses the OTP as the response. The request contains a displayable message. The OTP method is defined in RFC 2289. The OTP mechanism is employed extensively in VPN and PPP scenarios but not in the wireless world.

2.3.3 EAP-GTC

The EAP-GTC (Generic Token Card) is similar to the EAP-OTP except with hardware token cards. The request contains a displayable message, and the response contains the string read from the hardware token card.

2.4 Seven Problems on 802.11

2.4.1 Problem #1: Easy Access

Wireless LANs are easy to find. Strictly speaking, this is not a security threat. All wireless networks need to announce their existence so potential

clients can link up and use the services provided by the network. 802.11 require that networks periodically announce their existence to the world with special frames called Beacons.

However, the information needed to join a network is also the information needed to launch an attack on a network. Beacon frames are not processed by any privacy functions, which means that the 802.11 network and its parameters are available for anybody with an 802.11 card. "War drivers" have used high-gain antennas and software to log the appearance of Beacon frames and associate them with a geographic location using GPS.

Short of moving into heavily-shielded office space that does not allow RF signals to escape, there is no solution for this problem. The best you can do is to mitigate the risk by using strong access control and encryption solutions to prevent a wireless network from being used as an easy entry point into the network. Deploy access points outside firewalls, and protect sensitive traffic with VPNs.

2.4.2 Problem #2: "Rogue" Access Points

Easy access to wireless LANs is coupled with easy deployment. When combined, these two characteristics can cause headaches for network administrators. Any user can run to a nearby computer store, purchase an access point and connect it to the corporate network without authorization. Many access points are now priced well within the signing authority of even the most junior managers. Departments may also be able to roll out their own wireless LANs without authorization from the powers that be.

"Rogue" access points deployed by end users pose great security risks. End users are not security experts, and may not be aware of the risks posed by wireless LANs. Most existing small deployments mapped by war drivers do not enable the security features on products, and many access

points have had only minimal changes made to the default settings. It is hard to believe that end users within a large corporation will do much better.

Unfortunately, no good solution exists to this concern. Tools like NetStumbler allow network administrators to wander their building looking for unauthorized access points, but it is expensive to devote time to wandering the building looking for new access points.

Monitoring tools will also pick up other access points in the area, which may be a concern if you are sharing a building or a floor with another organization. Their access points may cover part of your floor space, but their access points do not directly compromise your network and are not cause for alarm. The periodic "walk-through" of your campus is the only way to address the threat of unauthorized deployment. At least network analyzers are moving to a handheld form, so you won't have to carry as much.

2.4.3 Problem #3: Unauthorized Use of Service

Several war drivers have published results indicating that a clear majority of access points are put in service with only minimal modifications to their default configuration. Nearly all of the access points running with default configurations have not activated WEP (Wired Equivalent Privacy) or have a default key used by all the vendor's products out of the box. Without WEP, network access is usually there for the taking.

Two problems can result from such open access. In addition to bandwidth charges for unauthorized use, legal problems may result. Unauthorized users may not necessarily obey your service provider's terms of service and it may take only one spammer to cause your ISP to revoke your connectivity.

Whether unauthorized use is a problem depends on the objectives of the service. For corporate users extending wired networks, access to wireless networks must be as tightly controlled as for the existing wired network. Strong authentication is a must before access is granted to the network.

If you have deployed a VPN to protect the network from wireless clients, it probably has strong authentication capabilities already built-in. Administrators can also choose to use 802.1x to protect the network from unauthorized users at the logical point of attachment. 802.1x also allows administrators to select an authentication method based on Transport Layer Security (TLS), which can be used to ensure that users attach only to authorized access points.

Not all networks, however, need to deploy ironclad user authentication. Theft of service was a major concern for connectivity providers in "hot spots" such as hotels and airports. After all, the business model was to charge for network access, so preventing unauthorized access was a business requirement. In the wake of the spectacular failure of some of the former big-name players like MobileStar, the hot-spot connectivity industry is experimenting with new business models.

Newer players in the market have based the business model on the idea that free wireless network access is an amenity that might draw guests and convention business. In this newer business model, user authentication is necessary only to ensure accountability. Authentication using a Web browser is a perfectly acceptable solution because it allows sessions to be identified and does not require specialized client software or a certain model of 802.11 network interface.

2.4.4 Problem #4: Service and Performance Constraints

Wireless LANs have limited transmission capacity. Networks based on 802.11b have a bit rate of 11 Mbps, and networks based on the newer

802.11a technology have bit rates up to 54 Mbps. This capacity is shared between all the users associated with an access point. Due to MAC-layer overhead, the actual effective throughput tops out at roughly half of the nominal bit rate. It is not hard to imagine how local area applications might overwhelm such limited capacity, or how an attacker might launch a denial of service attack on the limited resources.

Radio capacity can be overwhelmed in several ways. It can be swamped by traffic coming in from the wired network at a rate greater than the radio channel can handle. If an attacker were to launch a ping flood from a Fast Ethernet segment, it could easily overwhelm the capacity of an access point. Depending on the deployment scenario, it might even be possible to overwhelm several access points by using a broadcast address as the destination of the ping flood.

Attackers could also inject traffic into the radio network without being attached to a wireless access point. The 802.11 MAC is designed to allow multiple networks to share the same space and radio channel. Attackers wishing to take out the wireless network could send their own traffic on the same radio channel and the target network would accommodate the new traffic as best it could using the CSMA/CA mechanisms in the standard.

Large traffic loads need not be maliciously generated, either, as any network engineer can tell you. Large file transfers or complex client/server systems may transfer large amounts of data over the network to assist users with their jobs. If enough users start pulling vast tracts of data through the same access point, network access may resemble sucking molasses through a straw north of the Arctic Circle in January.

Addressing performance problems starts with monitoring and discovering them. Many access points will report statistics via SNMP, but not with the level of detail required to make sense of end-user performance complaints. Wireless network analyzers can report on the signal quality

and network health at a single location, but tools designed for wireless network administrators are only beginning to emerge.

The initial commercial wireless analyzer offerings were straightforward ports of their wired cousins; new products such as AirMagnet's handheld analyzer look like extremely promising additions to the wireless network engineer's toolkit. No enterprise-class wireless network management system has yet emerged. Some performance complaints could be addressed by deploying a traffic shaper at the point at which a wireless LAN connects to your network backbone. While this will not defend against denial of service attacks, it may help prevent heavy users from monopolizing the radio resources in an area.

2.4.5 Problem #5: MAC Spoofing and Session Hijacking

802.11 networks do not authenticate frames. Every frame has a source address, but there is no guarantee that the station sending the frame actually put the frame "in the air". Just as on traditional Ethernet networks, there is no protection against forgery of frame source addresses.

Attackers can use spoofed frames to redirect traffic and corrupt ARP tables. At a much simpler level, attackers can observe the MAC addresses of stations in use on the network and adopt those addresses for malicious transmissions.

To prevent this class of attacks, user authentication mechanisms are being developed for 802.11 networks. By requiring authentication by potential users, unauthorized users can be kept from accessing the network (Denial of service attacks will still be possible, though, because nothing can keep attackers from having access to the radio layer.).

The basis for the user authentication mechanism is the 802.1x standard ratified in June 2001. 802.1x can be used to require user authentication

before accessing the network, but additional features are necessary to provide all of the key management functionality wireless networks require. The additional features are currently being ironed out by Task Group I for eventual ratification as 802.11i.

Attackers can use spoofed frames in active attacks as well. In addition to hijacking sessions, attackers can exploit the lack of authentication of access points. Access points are identified by their broadcasts of Beacon frames. Any station that claims to be an access point and broadcasts the right service set identifier (SSID, also commonly called a network name) will appear to be part of an authorized network.

Attackers can, however, easily pretend to be an access point because nothing in 802.11 requires an access point to prove it really is an access point. At that point, the attacker could potentially steal credentials and use them to gain access to the network through a man-in-the-middle (MITM) attack.

Fortunately, protocols that support mutual authentication are possible with 802.1x. Using methods based on TLS, access points will need to prove their identity before clients provide authentication credentials, and credentials are protected by strong cryptography for transmission over the air.

Session hijacking will not be completely solved until the 802.11 MAC adopts per-frame authentication. Until that point, if session hijacking is a concern, you must deploy a cryptographic protocol on top of 802.11 to protect against hijacking.

2.4.6 Problem #6: Traffic Analysis and Eavesdropping

802.11 provides no protection against attacks that passively observe traffic. The main risk is that 802.11 does not provide a way to secure data

in transit against eavesdropping. Frame headers are always "in the clear" and are visible to anybody with a wireless network analyzer. Security against eavesdropping was supposed to be provided by the much-maligned Wired Equivalent Privacy specification.

A great deal has been written about the flaws in WEP. It protects only the initial association with the network and user data frames. Management and control frames are not encrypted or authenticated by WEP, leaving an attacker wide latitude to disrupt transmissions with spoofed frames.

Early WEP implementations are vulnerable to cracking by tools such as AirSnort and WEPCrack, but the latest firmware releases from most vendors eliminate all known attacks. The latest products go one step farther and use key management protocols to change the WEP key every 15 minutes. Even the busiest wireless LAN does not generate enough data for known attacks to recover the key in 15 minutes.

Whether you rely on WEP solely, or layer stronger cryptographic solutions on top of it is largely a question of risk management. The latest product releases have no known vulnerabilities. While that is some comfort, the same claim could have been made in July 2001 before release of the current generation of WEP-cracking tools. If your wireless LAN is being used for sensitive data, WEP may very well be insufficient for your needs. Strong cryptographic solutions like SSH, SSL, and IPsec were designed to transmit data securely over public channels and have proven resistant to attack over many years and will almost certainly provide a higher level of security. [6]

2.4.7 Problem #7: Higher Level Attacks

Once an attacker gains access to a wireless network, it can serve as a launch point for attacks on other systems. Many networks have a hard outer shell composed of perimeter security devices that are carefully

configured and meticulously monitored. Inside the shell, though, is a soft, vulnerable (and tasty?) center.

Wireless LANs can be deployed quickly if they are directly connected to the vulnerable backbone, but that exposes the network to attack. Depending on the perimeter security in place, it may also expose other networks to attack, and you can bet that you will be quite unpopular if your network is used as a launch pad for attacks on the rest of the world. The solution is straightforward in theory: Treat the wireless network as something outside the security perimeter, but with special access to the inside of the network. Although security diligence is time consuming, so is being sued.

CHAPTER 3

Wi-Fi NETWORK TRAFFIC

3.1 Connection / Access Protocol in Wi-Fi Networks

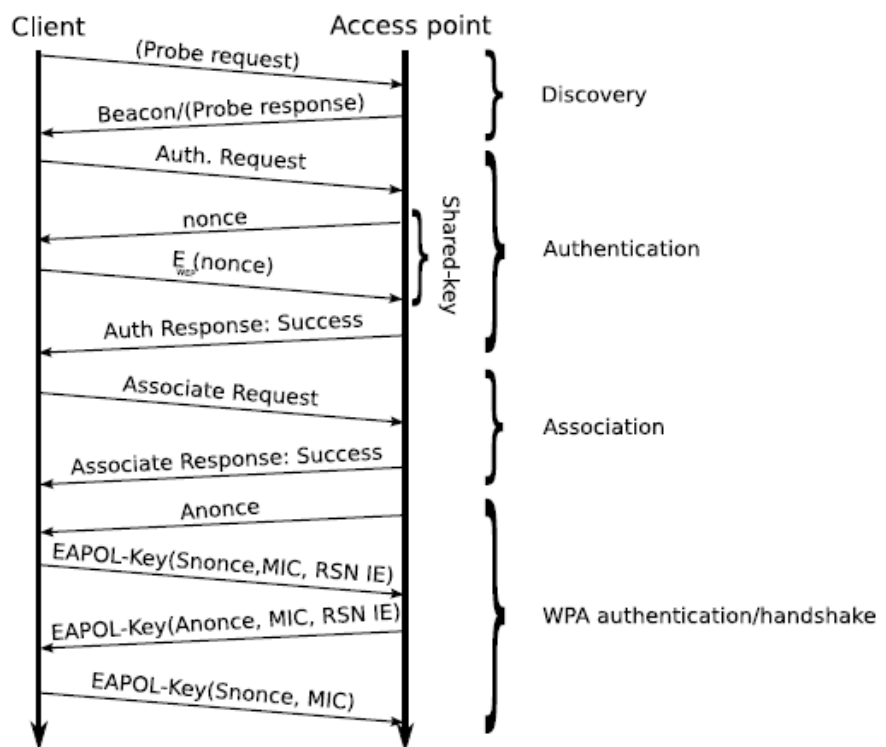


Figure 3 The protocols of connecting to a Wi-Fi network.

Figure 3 illustrates the basic protocols and flow of frames when connecting to an access point. First the client will detect access points either by sending a probe request and receiving a probe response, or purely by looking at the beacon frames frequently transmitted by an access point. Upon discovery, the client may try to authenticate to the access point. If successfully authenticated, the client may try to associate with the access point by sending an association request. If permitted by the access point the client will receive a positive association response. Whenever WPA is enabled, the shared-key authentication mechanism of WEP is skipped (open system authentication is used), and the real authentication is performed after association.

During this four-way handshake, important keys are generated and exchanged. After the initialization, the client is permitted and able to send and receive data frames to and from the network.

3.2 Analyzing Wi-Fi Network Traffic

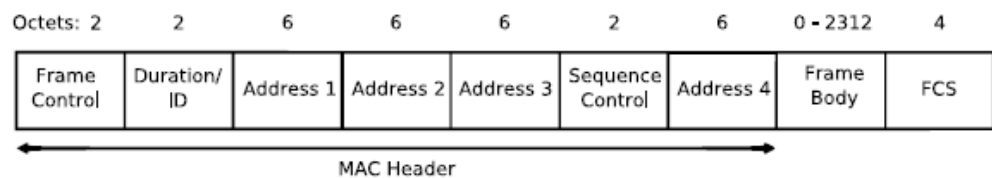


Figure 4 MAC frame format.

Every packet transmitted in Wi-Fi networks contain bits of information used to maintain the various layers of the communication. Although packets may be encrypted in Wi-Fi networks, they still have plaintext headers. As this section shows, the headers are valuable to anyone analyzing the network. The entire MAC frame displayed in Figure 4 is easily available to user-space tools in Linux². All packets in a Wi-Fi network conform to the MAC frame format. The Frame Control field specifies which type of payload the MAC frame transports. There are three main types of packets

² Put the interface into monitor mode and it will pass on the entire MAC frame to listeners.

and many subtypes. The main types, in bold and their subtypes, are:

1. Management: Association, Probe, Beacon, and Authentication.
2. Control: RTS, CTS, PS-Poll, ACK, CF-Ack/Poll.
3. Data: Data, Data + CF-Ack/Poll and Null-function.

In the following sections, only the interesting fields of interesting frames are discussed [7].

3.3 Information From All Frames

Figure 5 shows the frame control field. From it, the following information can be extracted.

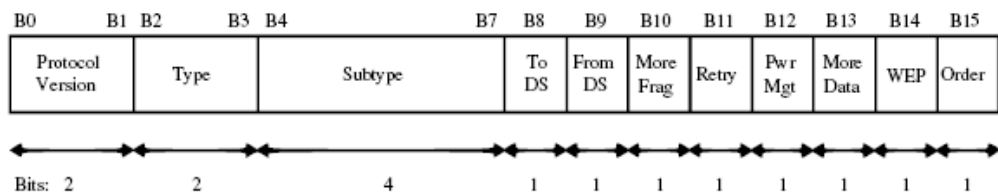


Figure 5 Frame control field.

Network is part of a WDS8: ToDS = 1 and FromDS = 1.

Network is in ad-hoc mode: ToDS = 0 and FromDS = 0; and Type = Data.

Network is in infrastructure mode: ToDS = 1 or FromDS = 1; and Type = Data.

Additionally, every captured frame includes signal-strength measured by the radio receiver. When combining this data with GPS-coordinates, it is possible to estimate:

Network range: Wherever frames from an access point were received.

Access point location: Triangulate from position and signal strength of frames transmitted by the access point and captured in multiple locations.

Client location: Same procedure as above, but only on frames transmitted from the desired client.

Buildings, other obstacles, and multipath fading will reduce the accuracy of the estimations. Moving clients or access points are not handled either and introduce errors. [8]

3.4 Information From Data Frames

WEP or WPA encryption: B14 = 1

Type of payload: E.g. if the destination address is the broadcast address, and the size of the payload is 68 bytes, then it is very likely to be an Address Resolution Protocol (ARP) request.

Network is a bridge: Only data packets with Frame Capability: ToDS = 1 and FromDS = 1 are transmitted.

MAC address of access point: In MAC header: Address 1, 2 or 3.

MAC address of mobile stations: In MAC header: Address 1, 2, 3 or 4.

Another piece that is valuable is the IV. It is sent with every data frame in an encrypted network. The IV and the use of makes it possible to guess from sniffed data frames exclusively, if the encryption scheme is WEP or WPA. When comparing frames from the same transmitting address, the IV is different with each frame for WEP. However, WPA has duplicate values in the 3-byte IV field several frames in a row, only the Extended Initialization Vector (EIV) values change for each field. [9]

The payload of the data frames can be ARP, Internet Protocol (IP) [10] Internet Control Message Protocol (ICMP) [11], Transport Control Protocol (TCP) [12], Universal Datagram Protocol (UDP), etc. All of these are

appended to Subnetwork Access Protocol (SNAP) [13] headers which are specific to ethernet. The different types of packets and knowledge of their structures are used in the next chapters to enable and improve some of the attacks described there.

3.5 Information From Management Frames

Some management frames transmit many parameters about the network. The beacon frame is one of them. Access points will broadcast beacon frames to inform stations that they are available. The frames provide enough information for a client to be able to join the network. However management frames are strictly used to administer the network connections. They do not send any data from the application layer. The capability field is part of the beacon frame. Its structure is depicted in Figure 6.

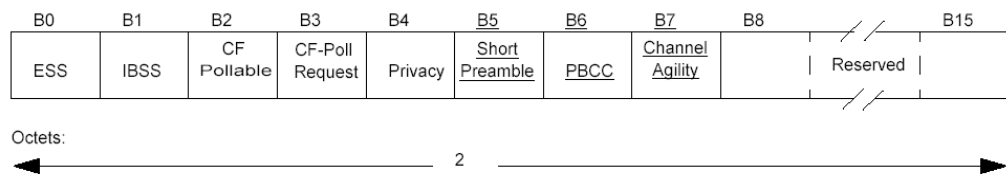


Figure 6 Capability field of the beacon frame.

From the capability field the following useful information can be extracted:

Network is in infrastructure mode: B0 = 1 and B1 = 0.

Network is in ad-hoc mode: B0 = 0 and B1 = 1.

WEP is required: B4 = 1.

Other fields that can be extracted from the frame body of a beacon frame are:

Beacon interval: is the time between each transmitted beacon frame (typically $100 \times 1024 \mu s = 100 \text{ms}$).

Service Set Identity (SSID): a string of maximum 32 bytes/characters that gives a human readable identification of a Wi-Fi network. It also serves another purpose to group together multiple access points to form a network of collaborating access points.

Supported rates: a "list" of supported transmit rates in the network.

Extended supported rates: other supported rates.

Channel: the channel the network is operating on.

The Basic Service Set Identifier (BSSID) discloses information about who manufactured the access point. The first 16bits of the BSSID can be looked up from an IEEE database [14].

3.6 Summary of Wi-Fi traffic

Available information from passively capturing ordinary Wi-Fi traffic is compiled in Table 1:

Table 1 Information available from an analysis of Wi-Fi frames .

Fact	Frame	Requirements
WDS	Data	1 frame
Ad-hoc/Infrastructure	Beacon/Probe/Data	1 frame
Network range	Any	3 frames and GPS
Client/Access point location	Any	3 frames and GPS
WEP	Beacon/Probe/Data	1 frame

Table 1 Information available from an analysis of Wi-Fi frames .

Fact	Frame	Requirements
WPA	Beacon/Probe/Data	1 frame
SSID	Beacon/Probe	1 frame
Access point MAC address	Any	1 frame
Client MAC address	Probe Request/Data	1 frame
Wired client MAC address	Data	1 frame
Contents of data	Data	Intelligent guess

CHAPTER 4

WEP OVERVIEW

IEEE 802.11 defines a mechanism for encrypting the contents of 802.11 data frames. This scheme uses five elements directly relevant to its analysis:

- a) A key shared between all the members of the BSS (there are really four shared keys, but this is irrelevant to the analysis).
- b) An encryption algorithm. For WEP this is the RC4 stream cipher, used to generate a key stream, which is XORed against plaintext to produce cipher text.
- c) The corresponding decryption algorithm. For WEP this is the same as the encryption algorithm. RC4 is used to generate a key stream, which is XORed against the cipher text to reproduce plaintext. [15]
- d) A 24-bit initialization vector or IV. WEP appends the IV to the shared key; WEP uses this combined key and IV to generate the RC4 key schedule. WEP selects a new IV for every packet.
- e) An encapsulation that transports the IV and the cipher text from the sender (encryptor) to the receiver (decryptor).

- f) WEP also uses a CRC of the frame payload plaintext in its encapsulation. The CRC is computed over the data payload and then appended to the payload before encryption. WEP encrypts the CRC with the rest of the data payload.

The operation of WEP is very simple to describe. First, each member of the BSS is initialized with the shared key via an unspecified, implementation specific, out-of-band mechanism.

To send a WEP encapsulated frame, the sender calculates the CRC of the frame payload and appends it to the frame. It then selects a new IV, appends this to the shared key to form a “per-packet” key, and uses the result to generate an RC4 key schedule. The sender then uses RC4 to generate a key stream equal to the length of the frame payload plus CRC. The sender XORs the generated key stream against the plaintext payload data and CRC. The sender also inserts the IV into the appropriate field in the frame header, and sets a bit indicating this is a WEP encrypted packet. At this point, the WEP encapsulation is complete, and the frame can be sent to the peer.

To process a WEP frame, the receiver checks the “encrypted” bit in the arriving frame. If it is set, the receiver extracts the IV from the frame, appends it to the BSS shared key, and generates the “per-packet” RC4 key schedule. RC4 is applied to the key schedule to produce a key stream the length of the packet’s encrypted payload. The receiver then XORs this key stream against the encrypted payload to extract plaintext. Finally the receiver verifies the CRC of the decrypted payload data to verify that the frame data correctly decrypted.

Several features of this design require comment.

The first is that the loss of a single bit of a data stream encrypted under RC4 causes the loss of all the data following the lost bit. This is because

data loss desynchronizes the RC4 encryption and decryption engines. The resynchronization problem gets only worse as more bits become lost. Since 802.11 often drops entire packets, it is infeasible to use a stream cipher like RC4 across 802.11 frame boundaries. To be useful across packet boundaries, this environment instead requires that the cipher support a random access, “seek” type capability, where it is feasible to instantly and efficiently switch the cipher to any selected point in the key stream. Many stream ciphers offer random access support—any block cipher such as AES [16] in counter mode, for instance, or SEAL [17] for another. Instead of selecting a stream cipher with characteristics needed for a datagram environment, however, the WEP architecture accommodates itself to loss by reinitializing the cipher key schedule on every data frame.

Stream ciphers have a second property that is particularly important to the analysis: it is unsafe to use the same key twice, ever. Suppose the cipher produces a key stream of bits

$$k_1 k_2 k_3 \dots$$

The encryptor uses the key stream sequence to encrypt the plaintext stream $p_1 p_2 p_3 \dots$ into a cipher text stream $c_1 c_2 c_3 \dots$ by XORing each plaintext bit with the corresponding key stream bit:

$$c_i = p_i \oplus k_i, \text{ for } i = 1, 2, 3, \dots$$

(Most practical implementations actually XOR whole bytes or words instead of bits.) The decryptor recovers the plaintext stream from the ciphertext stream by XORing each ciphertext bit with the corresponding key stream bit:

$$p_i = c_i \oplus k_i, \text{ for } i = 1, 2, 3, \dots (*)$$

The cipher stream is public knowledge, and it is presumed that adversaries will record the entire stream. If an adversary learns the plaintext value of bit i , she can recover the corresponding plaintext value of any other ciphertext stream encrypted from the same key stream: first compute the key stream bit

and then use equation (*) to decrypt the corresponding bit of any other stream encrypted under the same key.

$$k_i = c_i \oplus p_i$$

The WEP design attempts to accommodate this second property by introducing the IV. WEP combines the IV with the key to produce a new frame specific encryption key [18].

4.1 Decrypting Data Without Keys

This clause begins by describing problems with the WEP IV. Then it turns to practical techniques to recover the WEP RC4 key stream, based on the WEP IV deficiencies. Finally it closes with a discussion of some issues around the analysis [19].

4.2 WEP IV Problems

The WEP IV is 24 bits long. WEP appends the IV to the shared key to form a family of 2^{24} keys. As described above, each frame transmission selects one of these 2^{24} keys and encrypts the data under the key.

This scheme suffers from a basic problem. Since a stream cipher key stream can never be reused, it obliges the BSS to change the base key as soon as its members have consumed all of the 2^{24} keys derived from the base key. WEP defines no practical way to accomplish this, so in practice WEP keys are not replaced frequently enough to maintain the level of

privacy intended. This leads to wide-spread key abuse; a single access point BSS running at 11 Mbps and with a typical packet distribution can exhaust the derived key space in about an hour. A multi-access point network with tens or hundreds or thousands of access points can exhaust the key space at a faster rate, indeed, inversely proportional to the number of access points.

The problem is worse than this suggests, however. Since WEP shares the same base key among all the members of the BSS and since the security of WEP depends on the <base-key, IV> pair never being recycled, WEP needs an IV avoidance algorithm, to prevent one node from reusing an IV already used by another. WEP defines no such algorithm and it is unclear how to even begin to design one. A BSS could, for instance, partition the IV space among the BSS elements in a pre-defined manner, but this sort scheme either pre-supposes a static BSS membership static behavior or some (secure) scheme to transfer an indication of which IVs have been used among members of the BSS, etc.

The usual way to avoid this kind of difficulty is to randomly select the IV instead. Random selection of the IV, however, presents its own difficulties because of the Birthday Paradox (see [20] or [21]). The Birthday Paradox is named for the counter-intuitive fact that, in a group of people as small as 23, there is a 50% chance that two members of the group will share the same birthday. In general, if a set has n members, and elements are selected from the set one at a time with replacement, then the probability of a duplicate after two draws is $p_2 = 1/n$ and, for $k \geq 3$, the probability of at least one duplicate is

$$p_k = p_{k-1} + (k-1) \cdot 1/n \cdot (1 - p_{k-1}).$$

In the WEP case the IV space takes $n = 2^{24}$, and we exceed a 50% chance of a collision among IVs after only $k = 4823 \cdot 2^{12}$ frames. The probability of collision is already 99% after 12,430 frames, or in 2 to 3

seconds of normal traffic at 11 Mbps. There is already a 10% chance of collision after 1881 frames, a 1% chance after 582, a 0.1% after 184 frames, 0.01% after 59, and 0.001% after only 19 frames. With randomly selected IVs, maintaining the five zeroes of assurance (0.000001%) becoming customary in many fields of computing requires changing the base key after all the members of a BSS have transmitted a total 6 frames under the key! The odds are a normal 11 Mbps BSS will begin to reuse keys in less than a second of operation and there is a non-negligible probability that an attack can succeed well before this time has elapsed. The WEP IV space is far, far too small to protect against IV abuse.

It is important to be clear on what this does not say. It does not say that 50% of the IVs (and hence keys) will collide at about 2^{12} packets. It says that if an adversary collects a packet trace of about 2^{12} frames, there is about a 50% chance that the trace will contain at least one duplicate IV. But this is all the help an attacker needs.

4.3 Some Attacks

So how does an attacker exploit the key reuse brought about by WEP's IV duplication? There is no magic here. All of the attacks are standard. WEP does not protect against any of them.

As with any stream cipher, an attacker can launch known plaintext and chosen ciphertext attacks. In today's computing environments, these attacks are ridiculously easy. The attacker Eve, for instance, can forge e-mail from Bob, asking the victim Alice to e-mail a file with known content. A telephone call from Eve works just as well, if Alice knows Eve or Eve can pass herself off as Bob. If Eve does not know Alice, she can use an e-mail anonymizer to send spam to her. All of this appears completely innocent, but it causes known text to be encrypted. Eve captures the encrypted text with a packet sniffer and uses the relation

$$k_i = c_i \oplus p_i$$

to recover the key stream. She can save any key streams generated this way, each indexed by the appropriate IV, and immediately decrypt any WEP frame she sees later (or appears earlier in the packet trace!) with the same IV.

Eve can work a little harder, with two sniffers, one on the radio link and one outside the firewall of the organization being attacked. Using a bit of traffic analysis, it is not difficult to correlate the unencrypted traffic recorded from outside the firewall with encrypted traffic captured from the over-the-air link. She repeats the same process above to recover more key streams.

When 802.11 is used as the data link for a TCP/IP network, every data frame contains an IP datagram conveying large amounts of known plaintext. In such an environment Eve can recover a partial key stream for *every* frame sent. Traffic analysis can usually identify the type of traffic fairly accurately even when it is encrypted, and this information can be used to guess the values of variable fields in the packet headers, such as IP addresses and protocol numbers. This reveals even more of the data and hence key stream. The known plaintext from a single DHCP exchange can provide sufficient information to decode almost the entire TCP/IP header of every subsequent IP datagram encrypted by the DHCP client. Similarly, any “decrypted” packet can provide context and hints to help identify the plaintext of a still not fully decrypted packet, and this process can continue until the key streams are revealed for almost every IV.

As easy as these attacks are, they are largely for amateurs; there is no need to work this hard; at 24 bits, collisions come to you rapidly, one on the order of every second or two. A serious attacker will simply trace the 802.11 frames and XOR together ciphertext streams produced under the same IV. If an IV is used at least twice, then the packet trace will show the cipher streams

$$c_i = p_i \oplus k_i, \text{ for } i = 1, 2, 3, \dots$$

$$c_i \oplus = p_i \oplus \oplus k_i, \text{ for } i = 1, 2, 3, \dots$$

Produced from the key stream $k_1 k_2 k_3 \dots$ associated with the IV. XORing these together yields

$$p_i \oplus \oplus p_i, \text{ for } i = 1, 2, 3, \dots$$

i.e., the attacker knows with certainty the XOR of corresponding plaintext bits from $p_1 \oplus p_2 \oplus p_3 \dots$ and $p_1 p_2 p_3 \dots$. This means the attacker can know when each bit is the same or different in the two streams, so can immediately reduce the number of possibilities for each byte pair $\langle b, b \oplus \rangle$ from the two streams from 2^{16} possibilities to 2^8 . Having a third or a fourth frame encrypted under the same IV can reduce the possibilities for the two plaintexts even more. These facts mean pattern recognition techniques can often split apart the intertwined plaintext streams.

For instance, in any computing system a considerable percentage of actual data exchanged over a link is ordinary text from some natural language. In any natural language represented by an alphabet, certain character sequences occur more frequently than others, and the probabilities for various character sequences have been computed from empirical measurement. These facts mean that with a fairly high probability the adversary can easily and mechanically eliminate all but one of the 2^8 possibilities for almost all of the byte pairs $\langle b, b \oplus \rangle$. The WEP checksum can be used to referee among guesses for the few that cannot be eliminated by probabilities. In this way the attacker can recover the plaintext and the key stream without knowing any of the plaintext in advance. There is a good reason why the operating instructions for RC4 explicitly warn never to use the same key twice! [22]

As already noted, WEP as constituted today has no automated mechanism to change the BSS key, so people change it only infrequently,

typically on the order of days or weeks or even months. After n hours of typical use, it is likely that the overwhelming number of IVs have been used at least $n/2$ times, so a packet trace of all the BSS traffic provides collisions among most of the IVs. In this way, an attacker can easily build up a dictionary of key streams for every IV. Since WEP frames are small, and since there are only 2^{24} possible IVs, an attacker using even obsolete hardware can afford to store the key streams for all the IVs associated with thousands of base keys. Given that people tend to cycle through a small number manually configured base keys, an attacker usually only has to resort to these techniques only a few times before the BSS's privacy is permanently compromised. It is simply not worth trying to provide any privacy if this is the best we can do.

There is still one more problem with WEP's use of the IV that is worth noting. As we have seen, it is feasible to recover the key streams associated with a large number of IVs in a short time. It is mechanical to use RC4's definition along with the key stream and plaintext to reconstruct the underlying key schedule. Not every recovered key stream will reconstruct the entire key schedule, but we can expect to recover a certain amount of these. When we have recovered a full key schedule, it is worth asking ourselves: can we also recover the key used to compute it? We know the IV corresponding to the key schedule is the least significant 24 bits of the key, and we know the deterministic algorithm generating key schedule from the key. Can we compute the key using this knowledge? Questions like this worry cryptographers. WEP's highly questionable method of concatenating the IV directly to the base key exposes the base key to direct attack. Because of this, it would also be advisable to find a new way to mix the IV with the base key.

4.4 Problems With RC4

Most of the analysis thus far applies to any stream cipher, although we have noted a few problems with RC4 that makes its use by WEP a

dubious decision, the most important being it has no random access capability. RC4 also suffers from one more problem that makes its applicability to 802.11 even more questionable.

One in every 256 RC4 key is “weak” [23], which means the key schedules for these keys are less correlated with the key than they ought to be. This makes it far easier to cryptanalyze data encrypted under these keys, in case we fix enough of the other WEP problems to justify its continued use in new submissions. If WEP were to continue to use RC4, there is a standard fix to this problem that should be incorporated into WEP’s definition. This is to run the RC4 key stream generator at least 256 steps each time its key schedule is re-initialized, i.e. to begin with encrypting the 257th byte of the key stream instead of the first. Given that WEP already has to reinitialize the RC4 key schedule with every frame, this additional overhead probably costs too much to bear.

One final problem should be noted but is not a concern. The key stream RC4 produces exhibits an empirically measurable bias, meaning it only imperfectly hides correlation in the encrypted data. This problem is inconsequential as long as WEP uses only an infinitesimal part of the generated key stream (there is not sufficient data to correlate, since WEP recomputes the key schedule on every frame) and far easier attacks against WEP’s use of RC4 exist [24].

4.5 Cipher and Mode of Operation

All new symmetric key encryption efforts starting now should be based on the AES block cipher. It is thought to be as good as any symmetric key cipher in the public domain, and allows for very efficient implementations over a very wide range of environments (8-bit processors to super computers). This submission recommends against RC4 for any future WEP encapsulation scheme; attempts to twist WEP around RC4 have led to enough problems already. The reformulated WEP should instead

employ 128-bit AES as the mandatory to implement cipher.

This submission also recommends that WEP use AES in Offset Codebook Mode (OCB, [25]). This is a stream cipher that also produces a message authentication code, preventing an adversary from forging messages. The data encrypted under OCB is the same length as the plaintext data, a single key is used for both encryption and authentication, and it requires only the AES encryption, not decryption, engine. It is also a parallelizable mode, allowing for very high throughput. OCB has been optimized to minimize the number of calls to lower level cryptographic primitives and can both encrypt/decrypt and tag/verify a message in a single pass [26].

The OCB state is the key, a stride, which provides the offset in the mode's name and an IV. The stride and the IV are of the cipher block size (128 bits for AES). The stride is computed once per session. The per-frame overhead of OCB is 128 bits for the IV, and 128 bits for the OCB authentication tag [27].

4.6 Session Key Derivation

This submission recommends a session key derivation algorithm in the case of a manually configured base key, as used by WEP today. It does not recommend an algorithm for session key derivation when dynamic keying is available, because the scheme should incorporate state from the dynamic keying operation, to tie the key to the particular session that negotiated the key.

This algorithm produces two session keys, one for sending and the other for receiving.

1. Concatenate the (a) BSSID, (b) the sender's MAC address, and (c) the receiver's MAC address to produce a string. The order is important, as the two MAC addresses are reversed for sending and receiving.

- Using the base key (manually configured key) and an IV of 128 zero bits, run the OCB-AES algorithm on the concatenated string. The session key is the authentication tag output by this:

$session-key \oplus OCB-AES-tag_{base-key}(0, BSSID | sender-mac-addr | receiver-mac-address)$

Here 'a | b' means the concatenation of strings a and b.

The motives for this algorithm are (a) to remove the base-key from direct attack and (b) weakly tie the session key to the particular parties using it. Under this algorithm different sets of peers use different session keys, even though all the members of the BSS share the same base key. Note that the keys produced by this algorithm are still subject to dictionary attack when the base key is a password or derived from a password by techniques such as PKCS #5³. In addition, all the keys are subject to spoofing if the base key is revealed to an adversary. There is no magic that can avoid these weaknesses [28].

³ PKCS #5: Password-Based Encryption Standard, RSA Laboratories.

CHAPTER 5

Wi-Fi PROTECTED ACCESS (WPA)

In this section some of the security mechanisms of Wi-Fi Protected Access are given shortly. The few vulnerabilities inherent in WPA are demonstrated.

5.1 Background Information for WPA

5.1.1 WPA-PSK

Wi-Fi Protected Access—Pre-Shared Key (WPA-PSK) is currently the most common mode of operating a WPA protected Wi-Fi network. Much like WEP, a secret key is shared among all the clients in the network. This shared master key is called the Pairwise Master Key (PMK). When a client connects to an access point, a Pairwise Transient Key (PTK) is derived from the PMK, client and access point MAC address and a pair of nonces. From the PTK a MIC key is generated, which will be used to create MICs on the transmitted data. Also calculated from the PTK are the RC4 encryption keys, which are different from each encrypted frame.

5.1.2 Breaking Confidentiality

So far, only one attack to break the confidentiality provided by WPA is known. It uses the fact that a WPA key is often generated from a passphrase. By capturing the 4-way handshake of WPA authentication, an offline dictionary attack can be mounted.

CHAPTER 6

BREAKING THE SECURITY OF Wi-Fi

6.1 Recovering a Passphrase Seeded WPA Key

For security modes to be enabled in a user friendly manner, the secret PMK is often generated by a user supplied passphrase. The passphrase needs to be typed into the access point and each and every client that connects to the network. The function (Equation 1) to generate the PMK is openly available and is taken from [29]. The input is the passphrase, the SSID, length of the SSID, 4096 which specifies the number of times the algorithm should iterate, and 256—the size to output.

$$\text{PMK} = \text{PBKDF2}(\text{passphrase}, \text{ssid}, \text{ssidLength}, 4096, 256) \quad (1)$$

In order for a dictionary attack to be possible, it is necessary to validate if the PMK that is generated, is the correct key. With the help of the MIC this is possible. A captured packet is decrypted using the guessed PMK and a new MIC is generated over the decrypted data, with the MIC key from the guessed PMK. The original and newly generated MICs are compared and if they match the guessed PMK is likely to be the correct PMK [30].

WPA Cracker was the first tool to implement the offline dictionary attack against WPA. Its performance is approximately 24 passphrases per second when measured on a “AMD Athlon(tm) 64 Processor 2800+”. This tool requires the nonces, SSID and traffic dump of the handshake be inserted manually at start-up.

The popular tool Aircrack eventually implemented the WPA dictionary attack in addition to its powerful WEP attacks. A Pentium M processor running at 1.86 GHz manages to guess up to 150 passphrases per second, or use roughly one hour to check all the words in a Turkish or English word list.

Any word that may be found in a word list is a bad choice for a passphrase. Creating more words that match the usual requirements of a passphrase may be tried after going through the normal word lists. For instance, append numbers or symbols to the end of words, even just 123, 666, or “!”. John the Ripper is a tool to automate the creation of such passwords from simple word lists.

It seems few people choose good passwords and then only for their “important” accounts. Certainly they don’t use their important passwords to register on various on-line services such as forums. As many Wi-Fi routers are configured from the browser, there is a good chance they will choose a poor password since it is typed into the web browser.

```
# airodump ath0 dump
BSSID CH MB ENC PWR Packets LAN IP / # IVs ESSID
00:12:17:49:D1:81 6 48 WEP 21 23 0 linksys
00:13:10:9 B:47:F1 1 48 55 1279 118 Profelis
```

Listing 1 Airodump capturing the 4-way handshake.

6.2 WI-FI Protected Access (WPA)

In Listing 1 Airodump [31] will capture all traffic from the ath0 network interface, including the 4-way handshake after a client has associated. The traffic is stored in the file dump.cap.

```
# ./aireplay -O 5 -a 00:13:10:9B:47:F1 ath0
Use -c to target a specific station .
16:01:04 Sending DeAuth to broadcast -- BSSID : [00:13:10:9 B:47:F1]
16:01:04 Sending DeAuth to broadcast -- BSSID : [00:13:10:9 B:47:F1]
16:01:05 Sending DeAuth to broadcast -- BSSID : [00:13:10:9 B:47:F1]
16:01:09 Sending DeAuth to broadcast -- BSSID : [00:13:10:9 B:47:F1]
16:01:12 Sending DeAuth to broadcast -- BSSID : [00:13:10:9 B:47:F1]
```

Listing 2 Aireplay injecting de-authentication frames

The command in Listing 2 will force a 4-way handshake by transmitting de-authentication frames to everyone connected to the network. The parameter -O 5 instructs airplay to send 5 de-authentication frames, -a 00:13:10:9B:47:F1 sets the BSSID address of the frames to the correct address, ath0 is the Wi-Fi interface to transmit on. Each new line displayed represents the de-authentication frame that was transmitted. Most of the time the de-authentication client will re-authenticate milliseconds later. When the 4-way handshake has been captured by airodump it is time to start aircrack. In Listing 3 aircrack will perform the offline dictionary attack on the WPA PMK. Everything it needs to test passphrases is in the 4-way handshake. After aircrack has tested 38,480 passphrases it found “cokolata” which was the passphrase used in the WPA secured Wi-Fi network. The PMK, and the PTK used in the connection is also displayed. The last line is the MIC key.

```
# ./aircrack -e Profelis -w ../Tools/turkish.dic dump .cap
Opening dump .cap
Read 1507 packets .
aircrack 2.2
[00:04:15] 38480 keys tested (68.21 k/s)
KEY FOUND ! [ cikolata ]
Master Key : 4A A1 6A 13 CF 7A C7 72 6D F3 95 AE 5F 57 43 58
51 5F 52 C3 05 7D A5 97 8C 6F B3 90 93 8B 5C 37
Transcient Key : 34 1D 01 3D F9 1D 44 1A 34 D1 6A DE 7B A8 91 45
4B 25 7A 91 F0 1E 38 61 AD 14 9E 32 15 92 EA 0B
1C E3 DA D9 EA E5 D3 CE 60 06 B1 BE 0F 57 C6 40
67 F2 B9 CB 54 24 CD 10 64 DB 44 65 4D D7 80 D1
EAPOL HMAC : 26 D1 7B 4A C0 88 D1 DA F0 89 73 E6 47 DE 36 60
```

Listing 3 Aircrack performing the dictionary attack on WPA

6.3 Software Tools

Software is equally as important as the hardware involved in capturing information about Wi-Fi networks. Software tools described in the following sections reduce the amount of effort needed for anyone to study a network. There are different tools for different purposes where some tools, such as Kismet, give a good overview over a network or several networks while others like Ethereal give details about every byte or even bit of a packet. Another tool called Ettercap makes it easy to follow connections, which consists of related packets.

Although as pointed out, it is possible to learn a great deal about a network from its beacon frames. However, an engineer simply listening for beacon frames will not gain much knowledge he didn't already know. An engineer will typically want to know from where it is possible to use and connect to the network. Since it is likely that a client can hear the access point but not the other way around, engineers will want to go as far as associating with the access point under a site-survey.

If a cracker associates with the access point he becomes much more exposed. The process of associating requires two-way communication. The second the cracker transmits packets, the attack has become “active”. It is well known that active attacks are much more dangerous for a hacker, he may even be located or examined.

6.3.1 Kismet

Kismet is the de facto software tool for wardrivers. It uses most of the information in Section 4 and gives the wardriver a simple and user friendly UI with an overview of detected access points.

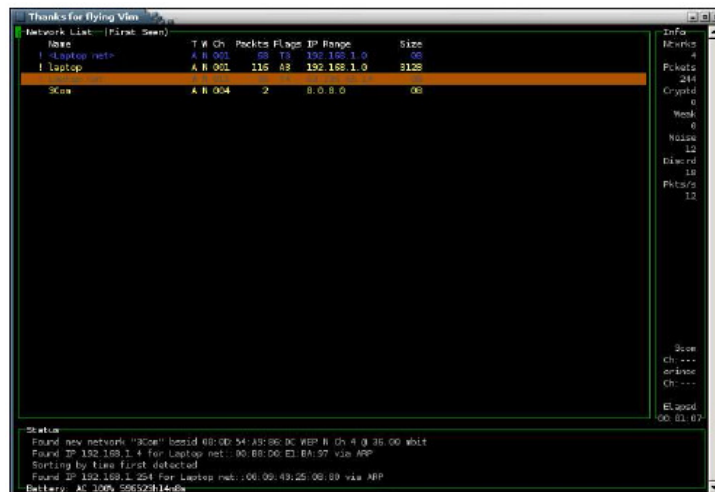


Figure 7 Kismet Screen Shot

In case visual feedback is difficult, come to mind warbikers and warwalkers, Kismet interfaces with a text-to-speech library and may inform its user of events via an earplug.

Kismet will communicate directly with the GPS receiver and record the position of every single received packet. This enables it to guess the physical location of the access point. An arrow in the user-interface tries to point the user in the right direction to the access point.

All currently available commercial Wi-Fi cards are restricted to listen on only a single channel at a given point in time. Kismet instructs the card to jump from channel to channel. It can also use two or more network cards to listen on multiple channels at a time. Kismet can be locked on to a particular channel to capture as much traffic from there as possible. Kismet compiles interesting statistics such as channel usage distribution and the percentage of WEP or WPA enabled networks.

Some access points disable broadcasting of their SSID in beacon frames or probe responses. Hiding the SSID is used to increase the security since only clients that know an access point's SSID are able to associate with it. But because management frames are transmitted in cleartext the SSID is also sent in cleartext when an "authenticated" client associates (authenticated in the sense that it has proved that it knows the "secret" SSID). Kismet will use this packet to display the network name of even so-called "hidden" or "cloaked" Wi-Fi networks.

6.3.2 TCPDump

TCPDump is an excellent tool to follow and filter communications in real time. In Listing, TCPDump listens on an associated Wi-Fi link for ARP packets. The TCPdump command is executed. Each line, apart from the first two, contains a description of the captured packet. The first packet captured is an ARP request, captured at 11 hours, 58 minutes 1.704626 seconds. The request is asking for who has the IP address 192.168.1.2 and to send the ARP response to 192.168.1.213. The request goes unanswered since nobody owns 192.168.1.2. Later the request for 192.168.1.213 is replied to by the network card with MAC address 00:0e:35:a3:0f:56 since that card owns the 192.168.1.213 IP address.

```
# tcpdump -l eth2 arp
tcpdump : verbose output suppressed , use -v or -vv for full protocol decode
listening on eth2 , link - type EN10MB ( Ethernet ) , capture size 96 bytes
11:58:01.704626 arp who - has 192.168.1.2 tell 192.168.1.213
11:58:02.704491 arp who - has 192.168.1.2 tell 192.168.1.213
11:58:03.704355 arp who - has 192.168.1.2 tell 192.168.1.213
11:58:44.184709 arp who - has 192.168.1.213 tell 192.168.1.1
11:58:44.184733 arp reply 192.168.1.213 is - at 00:0e:35:a3:0f:56 ( oui Unknown
)
```

Listing 4 Looking for ARP packets

6.3.3 Ethreal

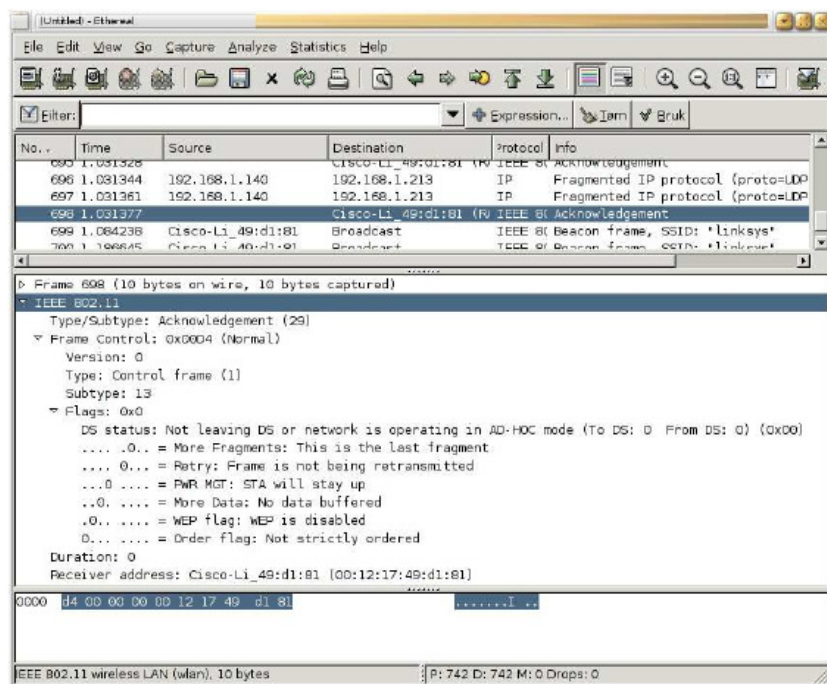


Figure 8 Ethreal Screen Shot

Ethereal is useful for an in depth look at a single packet including all available headers. Libraries over the structure of several types of packets give the user hints for each fragment of a packet. If Kismet does not provide enough feedback of what it finds, then Ethereal may be used to dig out the valuable parameters on the network or communication. Figure

is a screenshot of Ethereal. The first list in the window gives frames that are captured by the monitoring Wi-Fi network interface card. A dissection of an Acknowledgment frame is displayed in the list below. Fields of the frame are described in human-readable sentences or words. The last list in the window is the raw frame in hex-notation, and American Standards Character (ASCII) to the far right.

6.3.4 Ettercap

Ettercap is a user friendly packet sniffer. It gives a great overview over the connections in the network and may display the network traffic. Several plug-ins have been developed to do a number of attacks. In its current incarnation it

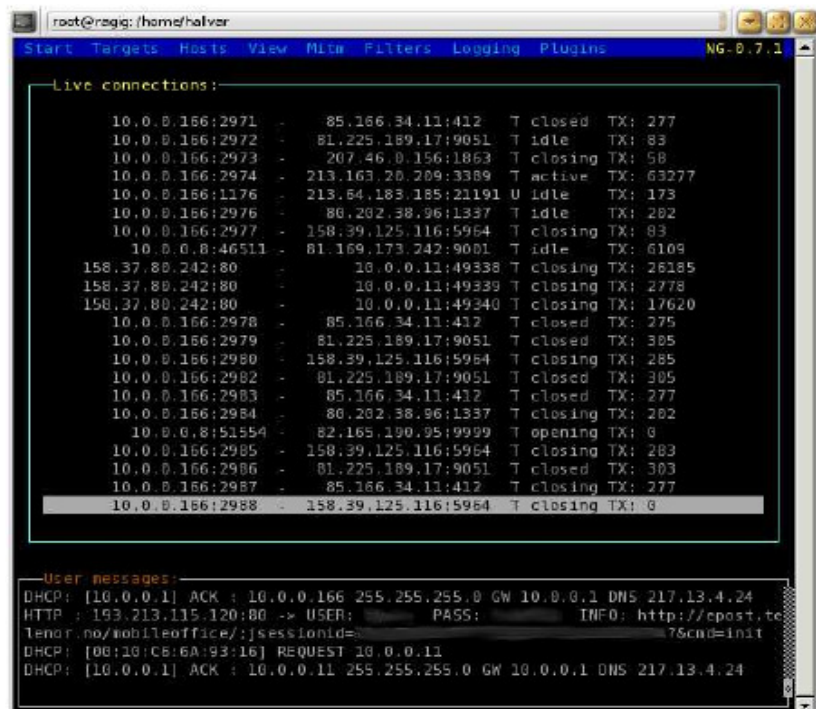


Figure 9 Ettercap Screen Shot

requires the network card to associate with the access point in order to show live network traffic. However it may display pre-captured data, even if it is WEP encrypted, although the WEP key must be provided to it. [32]

In Figure 9 a session with Ettercap is depicted. Ettercap can display a number of screens; in the figure it displays all live connections in the network with their source and destination IP address, and port numbers. The lower part of the screen is a console where interesting information that Ettercap finds will be displayed to the user.

6.3.5 IPTables

Iptables is Linux based firewall. You can make network address translation and masquerading using iptables.

6.3.6 HostAP

HostAP is a Linux driver for wireless LAN cards based on Intersil's Prism2/2.5/3 chipset. The driver supports a so called Host AP mode, i.e. it takes care of IEEE 802.11 management functions in the host computer and acts as an access point. This does not require any special firmware for the wireless LAN card. In addition to this, it has support for normal station operations in BSS and possible also in IBSS. WPA and RSN (WPA2) are supported when used with accompanied tools, wpa_supplicant (WPA/RSN Supplicant) and hostapd (WPA/RSN Authenticator). All these programs have been designed for both desktop/laptop computers and embedded systems.

Intersil's station firmware for Prism2 chipset supports a so called Host AP mode in which the firmware takes care of time critical tasks like beacon sending and frame acknowledging, but leaves other management tasks to host computer driver. This driver implements basic functionality needed to initialize and configure Prism2-based cards, to send and receive frames, and to gather statistics. In addition, it includes an implementation of following IEEE 802.11 functions: authentication (and de-authentication), association (reassociation, and disassociation), data transmission between

two wireless stations, power saving (PS) mode signaling and frame buffering for PS stations. The driver has also various features for development debugging and for researching IEEE 802.11 environments like access to hardware configuration records, I/O registers, and frames with 802.11 headers.

6.3.7 WPA Supplicant

wpa_supplicant is a WPA Supplicant for Linux, BSD, and Windows with support for WPA and WPA2 (IEEE 802.11i / RSN). It is suitable for both desktop/laptop computers and embedded systems. Supplicant is the IEEE 802.1X/WPA component that is used in the client stations. It implements key negotiation with a WPA Authenticator and it controls the roaming and IEEE 802.11 authentication/association of the wlan driver.

6.3.8 MadWiFi

MadWifi is short for *Multiband Atheros Driver for Wireless Fidelity*. In other words: this project provides a Linux kernel device driver for Atheros-based Wireless LAN devices. The driver works such that your WLAN card will appear as a normal network interface in the system. Additionally there is support for the Wireless Extensions API. This allows you to configure the device using common wireless tools (ifconfig, iwconfig and friends).

CHAPTER 7

GSM LOCATION BASED AUTHENTICATION USING SMS

We know that GSM operators can get the location information from the stations. With this method we can get the users position other than using Wi-Fi.

The method has some requirements:

1. GSM station.
2. Mobile Phone with Mobile Sign.
3. Mobile Operator.
4. Wireless AP which is synchronized by mobile operator.
5. Client

The method is simply shown in figure.

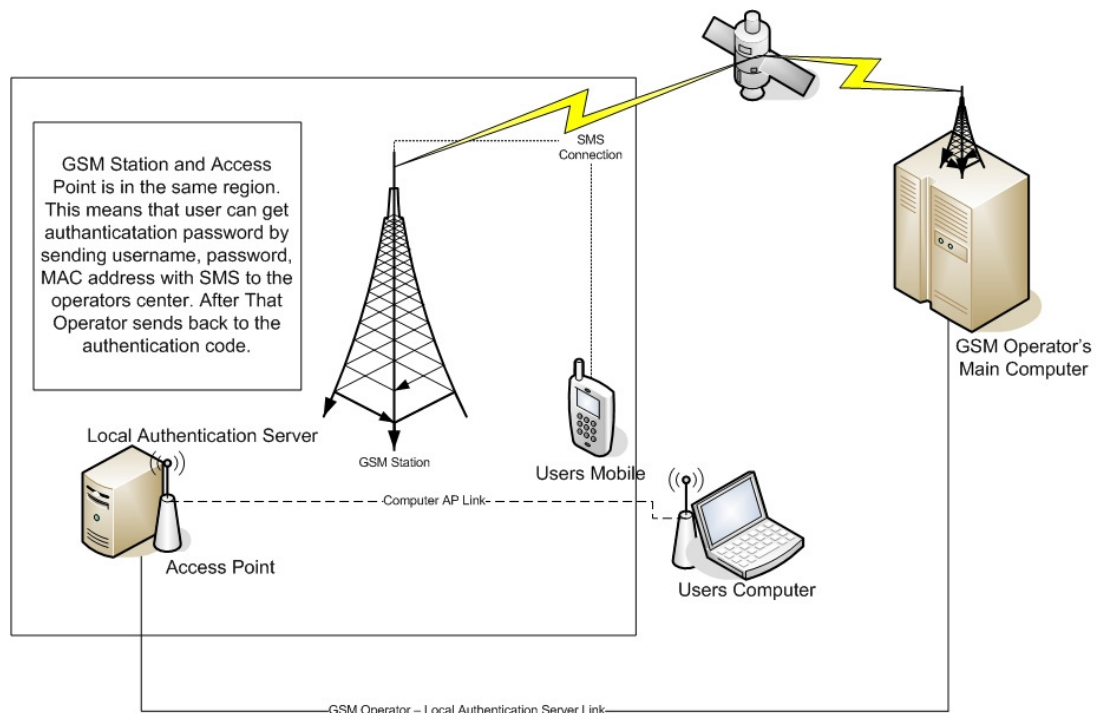


Figure 10 GSM Location Based Authentication using SMS

In Figure 10, the client wants to connect internet using its wireless card. Server must authenticate user. For security reasons user request must be logged. There are some commercial sites which shows your location on the internet site like: <http://www.mapamobile.com/>. For Turkey, TURKCELL⁴ has similar applications for monitoring vehicles on the map. Also the police is using such services for searching stolen mobile phones.

7.1 Steps of Authentication

Step 1: User request OTP from his/her mobile phone by sending SMS with mobile sign [33] in this format:

<MAC> <User Name> <Password>

Step 2: Operator receives the SMS and checks user name, password,

⁴ TURKCELL: One of the biggest mobile operator companies in Turkey.

MAC, location, mobile phone number. If all information has matched with the database, server sends back SMS which includes SSID and OTP in this format:

<SSID> <OTP>

Step 3: User authenticates himself/herself by using that information.

In this method EAP-OTP can be used or simply WAP or WEP can be used. Some codes are added to this thesis, which are related to this method as a proof on this concept. The code is examined using WPA-PSK. But if it can be EAP-OTP it is better for working model.

There is also an SMS server⁵ which can be used by the user without mobile operator.

With this technique, multiple technologies are combined in one hand. There is no new authentication method. There is just an authentication database from operator side. WPA-PSK can be changed by the time.

When user requests a SSID and password from operator the main problem is SMS sending and receiving time. The time value shall be set to the calculated time value.

⁵ Appendix B.4

CHAPTER 8

CONCLUSION

By using this method, mobile phone operators, hotels or some public area authenticate requests can be fulfilled. Also using these methods wireless attackers can be defeat.

As a feature work OTP JAVA can be developed for mobile phones. This key can be used with mobile sign.

In this thesis we achieve our goal. This method is useful for information technology law of Turkey service operator requirements. According to that law, operator must know the identity of who is connected to the network and all traffic information on the network.

REFERENCES

- [1] <http://www.aircrack-ng.org>

- [2] <http://ralink.rapla.net/>

- [3] <http://madwifi.org>

- [4] <http://www.kismetwireless.net>

- [5] **IEEE Std 802.11** (1997), *Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications*, IEEE.

- [6] **McGrew D. A. and Fluhrer S. R.** (2000), “*The stream cipher encapsulating security payload*”, draft-mcgrew-ipsec-scesp-01.txt, IETF.

- [7] **Helleseth H.** (2005), *Data set of WEP encrypted frames*, international WiFi conference, NY.

- [8] **Edney J. and Arbaugh A. W.** (2004), *Real 802.11 security, Wi-Fi protected access and 802.11i*. ISBN 0321136209.

- [9] **Dubrawsky I.** (2007), *Safe layer 2 security in-depth—version 2*, Printace Hall.

- [10] **Postel J.** (1981), *Internet protocol message protocol.*, RFC 792 (Standard).

- [11] **Postel J.** (1981), *Internet protocol message protocol*, RFC 791 (Standard).
- [12] **Postel J.** (1981), *Internet protocol message protocol*, RFC 793 (Standard).
- [13] **Postel J. and Reynolds J.K.** (1988), *A standard for the transmission of ip datagrams over ieee 802 networks*, RFC 1042 (Standard).
- [14] **IEEE Standards Association.**(2006) *IEEE OUI and company id assignments*, IEEE.
- [15] **Hulton D.** (2002), *Practical exploitation of RC4 weaknesses in WEP environments*, security conference of Canada'02.
- [16] **Scott Ananian C.**(2007) *Open source PPTP client*, open source PPTP client documentation.
- [17] **FortConsult Aps.** (2005) *Ny metode til at afsløre hackere*, FortConsult Web.
- [18] **Daemon J. and Rijmen V.** (2001), *AES proposal*, Rijndael.
- [19] **Menezes A. J., van Orschot P. C., and Vanstone S. A.** (1996), *Handbook of applied cryptography*, CRC.
- [20] **Roos A.** (1995), *A Class of weak keys in the RC4 stream cipher.*, CRC.
- [21] **Martin Scott Fluhrer I. and Shamir Ron Rivest A.**, *Weaknesses in the key scheduling algorithm of RC4*, Prentice Hall.

- [22] **Schneier B.** (1997), *Applied cryptography, 2nd addition*, Wiley.
- [23] **Eastlake D., Crocker S., and Schiller J.** (1994), *RFC 1750, Randomness recommendations for security*, IETF.
- [24] **Bellovin S. M.** (2000), "Problem areas for IP security protocols", 6th USENIX UNIX security conference.
- [25] **Rogaway P.** (2002), *OCB mode: parallelizable authenticated encryption*, UNIX security conference.
- [26] **Dierks T. and Allen C.** (1999), *RFC 2246, the TLS protocol, version 1.0*, IETF.
- [27] **Kent S. and Atkinson R.** (1998), *RFC 2401, security architecture for the internet protocol*, IETF.
- [28] **Rogaway P. and Coppersmith D.** (1994), "A software-oriented encryption algorithm", in *fast software encryption*, Cambridge security workshop, Springer-Verlag.
- [29] <http://www.rsasecurity.com/>
- [30] **Dingledine R. and Mathewson N.** (2006), *Tor protocol specification.*, IETF.
- [31] <http://www.dachb0den.com/projects/bsd-airtools>
- [32] **Helleseeth H.** (2005), *Data from warbiking in Bergen*, network security conference, Belgium.
- [33] http://www.turkcell.com.tr/bireysel/servisler/asistan/Turkcell_mobil_imza

APPENDIX A

Acronyms & Abbreviations

ACK	Acknowledgement
AP	Access Point
API	Application Programming Interface
ARP	Address Resolution Protocol
ASCII	American Standards Character
BS	Base Station
BSD	Berkeley Software Distribution
BSS	Basic Service Set
BSSID	Basic Service Set Identifier
CF	Contention-Free
CRC	Cyclic Redundancy Check
CTS	Clear To Send
dB	Decibel
dBi	Decibel Gain Related to an Isotropic Radiator
DCF	Distributed Coordination Function
DGPS	Differential Global Positioning System
DHCP	Dynamic Host Configuration Protocol
DIFS	Inter-Frame Spacing
DNS	Dynamic Name Resolution
DSSS	Direct Sequence Spread Spectrum
EIRP	Effective Isotropic Radiated Power
EIV	Extended Initialization Vector
ESS	Extended Service Set
ESSID	Extended Service Set Identifier

FCC	Federal Communications Commission
FCS	Frame Check Sequence
FMS	Fiat, M, and Shamir
FTP	File Transfer Protocol
GPS	Global Positioning System
HTTP	Hyper Text Transport Protocol
IBSS	Independent Basic Service Set
ICMP	Internet Control Message Protocol
ICV	Integrity Check Value
IP	Internet Protocol
IEEE	Institute of Electrical and Electronics Engineers
IMAP	Internet Message Access Protocol
IR	Infrared
ISM	Industrial, Scientific, and Medical
IV	Initialization Vector
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LLC	Link Layer Control
MAC	Medium Access Control
MadWiFi	Multiband Atheros Driver for WiFi
Mbps	Mega bits per second
MD5	Message Digest, version 5
MIC	Message Integrity Code
mini-PCI	Miniature Peripheral Component Interconnect
MPDU	MAC Protocol Data Unit
MPPE	Microsoft Point-to-Point Encryption
MS	Mobile Station
mW	Milli Watt
NAT	Network Address Translation
NMEA	National Marine Electronics Association

PAN	Personal Area Network
PCF	Point Coordination Function
PC-Card	Peripheral Component Interconnect Card
PCMCIA	Personal Computer Memory Card International Association
PGP	Pretty Good Privacy
PIFS	Inter-Frame Spacing
PKI	Public Key Infrastructure
PLCP	Physical Layer Convergence Protocol
PMK	Pairwise Master Key
PPTP	Point-to-Point Tunneling Protocol
PRGA	Pseudo Random Number Generator Algorithm
PRNG	Pseudo Random Number Generator
PRGN	Pseudo Random Number Generator Number
PS	Power Save
PSK	Pre-Shared Key
PTK	Pairwise Transient Key
QoS	Quality of Service
RC4	Rivest Cipher 4 or Ron's Code
RF	Radio Frequency
RSA	Rivest, Shamir, & Adleman
RSN IE	Robust Security Network Information Element
RSN	Robust Security Network
RTS	Request To Send
SNAP	Subnetwork Access Protocol
SSID	Service Set Identity
TCP	Transport Control Protocol
TCP/IP	Transport Control Protocol/Internet Protocol
TKIP	Temporal Key Integrity Protocol
U.FL	Highrose Connector
U-NII	Unlicensed National Information Structure

UDP	Universal Datagram Protocol
URI	Universal Resource Identifier
USB	Universal Serial Bus
VPN	Virtual Private Network
W	Watt
WDS	Wireless Distribution System
WEP	Wired Equivalent Privacy
Wi-Fi	Wireless-Fidelity
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access
WPA2	Wi-Fi Protected Access version 2
WPA-PSK	Wi-Fi Protected Access—Pre-Shared Key
XOR	Bitwise addition

APPENDIX B

Used Configurations

B.1. hostapd.conf

An additional configuration parameter, bridge,
must be used to notify hostapd if the interface is included in a bridge.

#bridge=br0 # Enable this for standard bridging, leave disabled for
netfilter firewalls

interface=ath0
driver=madwifi
logger_syslog=-1
logger_syslog_level=2
logger_stdout=-1
logger_stdout_level=2
debug=1
ctrl_interface_group=0
macaddr_acl=0
deny_mac_file=/etc/hostapd.deny
auth_algs=3
eapol_key_index_workaround=0
eap_server=0
dump_file=/tmp/hostapd.dump
ssid=caglar
wpa=3

```
wpa_psk=65edadd8c6a29b1e5a898ef9f9a18e4edb82982d7923565008aa  
a0213d13ea61  
wpa_key_mgmt=WPA-PSK  
wpa_pairwise=TKIP CCMP
```

B.2. wpa_supplicant.conf

```
network={  
    ssid="caglar"
```

```
    psk=65edadd8c6a29b1e5a898ef9f9a18e4edb82982d7923565008aaa021  
    3d13ea61  
    key_mgmt=WPA-PSK  
    proto=RSN  
}
```

B.3. Wireless_AP Configuration Shell Script

```
#!/bin/bash  
ifconfig ath0 down  
wlanconfig ath0 destroy  
wlanconfig ath0 create wlandev wifi0 wlanmode ap  
iwconfig ath0 mode Master  
  
echo -n "Enter ESSID of AP (Ex: caglar): "  
read AP_NAME  
iwconfig ath0 essid ${AP_NAME}  
  
echo -n "Enter IP for AP (Ex: 10.1.1.1): "  
read AP_IP  
ifconfig ath0 ${AP_IP} up
```



```
#firewall options
echo 1 > /proc/sys/net/ipv4/ip_forward
echo -n "Enter output interface (Ex: eth1): "
read AP_EXT_INT
iptables -t nat -A POSTROUTING -o ${AP_EXT_INT} -j MASQUERADE
iptables -F

clear
echo "DONE.."
iptables -t nat -L
iwconfig ath0
ifconfig ath0

hostapd -B /etc/hostapd.conf
```

B.4. SMS Server-Client Program

B.4.1. corePortAccess.c

```
/*
Port related functions are in this file.
version 2,stable.
*/

#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <termios.h>

/*
```

Nokia 7110 Connection Parameters

```
-----  
BAUD: 19200  
DATA BITS: 8  
STOP BITS: 1  
PARITY: NO PARITY  
FLOW CONTROL: H/W FLOW CONTROL (RTSCSTS)  
*/
```

```
int portInit(int mode, char *port)  
{  
    static int portDescriptor;  
    int status;  
    char whichPort[]="/dev/ttyS";  
    static struct termios oldSetup;  
  
    if(mode==1)  
    {  
        strcat(whichPort,port);  
        portDescriptor=open(whichPort, O_RDWR | O_NOCTTY);// |  
O_NDELAY); //Now Waits port reply  
        fcntl(portDescriptor, F_SETFL,0);// FNDELAY);  
        //setting port parameters  
  
        struct termios portSetup;  
  
        tcgetattr(portDescriptor, &oldSetup);  
        tcgetattr(portDescriptor, &portSetup);  
  
        cfsetispeed(&portSetup, B19200); //input baud  
        cfsetospeed(&portSetup, B19200); //output baud
```

```

portSetup.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG); //Raw
mode.
//portSetup.c_oflag &= ~OPOST; //Raw output
//portSetup.c_iflag &= ~OPOST; //Raw input

portSetup.c_cflag |= (CREAD | CLOCAL); // Read Enable, Local
Mode

portSetup.c_cflag &= ~PARENB; //Clear Parity bit (no parity)
portSetup.c_cflag &= ~CSTOPB; //Clear stop bit (1 stop bit)
portSetup.c_cflag &= ~CSIZE; //clear data size bits (0 data bits)
portSetup.c_cflag |= CS8; //set 8 data bits

portSetup.c_cflag |= CRTSCTS; //Set hardware flow control.

//portSetup.c_iflag |= (IGNPAR); //Ignore Parity.

tcsetattr(portDescriptor, TCSAFLUSH, &portSetup);
tcflush(portDescriptor, TCIOFLUSH);

return portDescriptor;
}

if(mode==0)
{
tcflush(portDescriptor, TCIOFLUSH);
tcsetattr(portDescriptor, TCSAFLUSH, &oldSetup);
close(portDescriptor);
}

return portDescriptor;
}

```

B.4.2. messageSendingCore.c

```
/*
SMS sending core. Used by other modules to send SMS when needed.
version 0.6alpha. //connecting to port, optimized a bit, nearly complete. /
stable
*/

#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <termios.h>

/*
Argument table for this program
-d or --destination : destination
-m or --message    : the text to be sent
-c or --center     : the message center to be used

Note: These arguments have no specific order.
*/

void help(char *name)
{
    printf("\ncommand line SMS sender core.\n");
    printf("-----\n");
    printf("Usage: %s [options]\n\n",name);
    printf("Options:\n");
    printf("\n-c --center [number]:Defines an SMS center to work with
<Optional>\n");
    printf("\n-d --destination [number]: Defines the destination of the
```

```

message.\n");
    printf("\n-m --message [\"message\"]: The message itself.\n");
    printf("\n-p --port [n]: defines which serial port to be used. Use port
number-1 for n. (i.e. 0 for com port 1).\n");
    printf("\n-i --interactive [0 | 1 | 2]: Debugging mode. 1 gives steps'
output. 2 enables confirmation.");
    printf("\n-h --help: This screen\n\n");
    printf("\nNote:Format the numbers of center and destination as you
are sending an ordinary SMS from your phone.");
}

```

```

int sms_main(int parameterCount, char * parameterValue[], int output)
{
    int interactiveMode=0;    //Interactive mode is used for debugging
purposes. if 1, prints information on screen.
                            //if 2, wants confirmation before sending.
    int portDescriptor;      //returning file descriptor from portInit()
function.
    int status;              //used for checking error conditions.

    int i,j;                 //classic loop constant.
    int counter;             //used in duplicate argument detector.

    int messageIndex=-1;    //Holds the array positions of variables
that are needed.
    int centerIndex=-1;
    int destinationIndex=-1;

    char *parameterMask[12]={"-c","--center","-d","--destination","-m", "--
message","-i","--interactive","-h","--help","-p","--port"};
    char centerBuffer[512];  //this is the variable that helps on editing
of message center.
    char destinationBuffer[512]; //this is the variable that helps on

```

editing of message destination.

```
char overflowPreventer[512]; //this variable must stay here. it's
acting as a buffer overflow green zone.
char *transferBuffer; //this is a temporary buffer for copying a
string into stringBuffer. (uses strcpy())
char *processBuffer; //another temporary buffer used for
adding components and for other temporary stuff.
char commandBuffer[512]; //a temporary buffer used for string
transfer to port.
char decision;
char *whichPort="0"; //used to look for which port is used.
```

```
FILE *f;
```

```
//checking for true formatting
```

```
if((parameterCount%2)==0)
{
    help(parameterValue[0]);
    return 0;
}
```

```
//Checking for too less parameters.
```

```
if(parameterCount<2)
{
    help(parameterValue[0]);
    return(0);
} //done.
```

```
//Checking for too many parameters.
```

```
if(parameterCount > 13)
{
    help(parameterValue[0]);
```

```

        return(0);
    }//done.

    //a duplicate argument detection mechanism
    for(i=0; i<12; i++)
    {
        counter=0;
        for(j=1; j<parameterCount; j++)
        {
            if(!strcmp(parameterValue[j],parameterMask[i]))
            {
                counter++;
            }
        }
        if (counter>1)
        {
            help(parameterValue[0]);
            return(0);
        }
    }//done.

```

```

    //searching for help request
    for(i=1; i<parameterCount; i++)
    {
        if(!strcmp(parameterMask[8],parameterValue[i]) ||
!strcmp(parameterMask[9],parameterValue[i]))
        {
            help(parameterValue[0]);
            return(0);
        }
    }

```

```

//searching for port number

```

```

for(i=1; i<parameterCount; i++)
{
    if(!strcmp(parameterMask[10],parameterValue[i]) ||
!strcmp(parameterMask[11],parameterValue[i]))
    {
        whichPort=parameterValue[i+1];
a    }
}

//searching for interactiveMode parameter
for(i=1; i<parameterCount; i++)
{
    if(!strcmp(parameterMask[6],parameterValue[i]) ||
!strcmp(parameterMask[7],parameterValue[i]))
    {
        interactiveMode=parameterValue[i+1];
    }
}

//Searching for message center

for(i=1; i<parameterCount-1; i++)
{
    if(!strcmp(parameterMask[0],parameterValue[i]) ||
!strcmp(parameterMask[1],parameterValue[i]))
    {
        if(interactiveMode)
        {
            printf("OK, message center found:");
        }

        centerIndex=i+1;

```



```

        if(interactiveMode)status=read(portDescriptor,
&processBuffer,255);
        {
            printf("%s\n",parameterValue[i+1]);
        }
    }
} //done

```

```

//searching for message destination
for(i=1; i<parameterCount-1; i++)
{
    if(!strcmp(parameterMask[2],parameterValue[i]) ||
!strcmp(parameterMask[3],parameterValue[i]))
    {
        if(interactiveMode)
        {
            printf("OK, message destination found:");
        }

        destinationIndex=i+1;

        if(interactiveMode)
        {
            printf("%s\n",parameterValue[i+1]);
        }
    }
} //done

```

//searching for message itself

```

for(i=1; i<parameterCount-1; i++)
{

```

```

        if(!strcmp(parameterMask[4],parameterValue[i]) ||
!strcmp(parameterMask[5],parameterValue[i]))
    {
        if(interactiveMode)
        {
            printf("OK, message found:");
        }

        messageIndex=i+1;

        if(interactiveMode)
        {
            printf("%s\n",parameterValue[i+1]);
        }
    }
} //done

//looking for too long messages.
if(strlen(parameterValue[messageIndex])>159)
{
    printf("Too long message. Please trim under 160 chars.\n");
    return 0;
}

//done.

//searching if vital parameters are here.
if(messageIndex==-1 || destinationIndex==-1)
{
    help(parameterValue[0]);
    return(-1);
}
//done.

```

```

//accessing "/dev/ttyS0"
portDescriptor=portInit(1,whichPort); // Setting up COM1 (/dev/ttyS0)
for accessing the phone.
if (portDescriptor==-1) // If cannot access, abort the execution.
{
    if(output)
    {
        printf("Cannot access device %c\n",whichPort);
    }
    else
    {
        f=fopen("/var/log/sms_server","a+");
        fprintf(f,"%s Cannot access device %c\n",zaman(),whichPort);
        fclose(f);
    }
    return(0);
}

if(interactiveMode)
{
    printf("Port descriptor obtained: %d\n", portDescriptor);
}

//string processing starts here

//first, message center.

if(centerIndex!=-1)
{
    processBuffer=parameterValue[centerIndex];
    transferBuffer="AT+CSCA=\\";
    strcpy(centerBuffer,transferBuffer);
}

```

```

        strcat(centerBuffer,processBuffer);
        processBuffer="\r";
        strcat(centerBuffer, processBuffer);
    }
//message center ends here.

//then, destination.
processBuffer=parameterValue[destinationIndex];
transferBuffer="AT+CMGS=\"";
strcpy(destinationBuffer, transferBuffer);
strcat(destinationBuffer, processBuffer);
processBuffer="\r";
strcat(destinationBuffer,processBuffer);
//destination ends here.
//end of string processing.

if(interactiveMode)
{
    printf("\nText re-formatting complete, results:\n");
    printf("-----\n");
    if(centerIndex!=-1)
    {
        printf("Center: %s\n", centerBuffer);
    }
    printf("Destination: %s\n", destinationBuffer);
    printf("Message: %s\n", parameterValue[messageIndex]);
}

if(interactiveMode>1)
{
    printf("\nLast exit before the bridge...\n");
    printf("-----\n");

```

```

printf("Send? [Y/n]:");
scanf("%c",&decision);
if(decision=='n' || decision=='N')
{
    printf("Aborted...\n");
    portInit(0);
    return 0;
}
}

//sending message.

if(interactiveMode)
{
    printf("\nStarting to send commands to phone\n");
    printf("-----\n");
}

sleep(1); //remove me. after testing of course.
transferBuffer="ATZ\n";
strcpy(commandBuffer,transferBuffer);
status=write(portDescriptor, commandBuffer,strlen(commandBuffer));

sleep(1);
if(interactiveMode)
{
    printf("Obtained port descriptor: %d\n", portDescriptor);
    printf("Write status for %s: %d\n",transferBuffer, status);
    status=read(portDescriptor, &processBuffer,50);
    printf("Read status for %s: %d\n", transferBuffer, status);
}
}

```

```
transferBuffer="ATE0\r";
strcpy(commandBuffer, transferBuffer);
status=write(portDescriptor, commandBuffer,strlen(commandBuffer));
```

```
sleep(1);
if(interactiveMode)
{
    printf("Write status for %s: %d\n",transferBuffer, status);
    status=read(portDescriptor, &processBuffer,50);
    printf("Read status for %s: %d\n", transferBuffer, status);
}
```

```
transferBuffer="AT+CMGF=1\r";
strcpy(commandBuffer,transferBuffer);
status=write(portDescriptor, commandBuffer,strlen(commandBuffer));
```

```
sleep(1);
if(interactiveMode)
{
    printf("Write status for %s: %d\n",transferBuffer, status);
    status=read(portDescriptor, &processBuffer,50);
    printf("Read status for %s: %d\n", transferBuffer, status);
}
```

```
if(centerIndex!=-1)
{
    status=write(portDescriptor, centerBuffer,strlen(centerBuffer));
```

```
sleep(1);
if(interactiveMode)
{
    printf("Write status for %s: %d\n", centerBuffer, status);
```

```

        status=read(portDescriptor, &processBuffer,50);
        printf("\nRead status for %s: %d\n",centerBuffer, status);
    }
}

    status=write(portDescriptor, destinationBuffer,
strlen(destinationBuffer));

    sleep(1);
    if(interactiveMode)
    {
        printf("Write status for %s: %d\n", destinationBuffer, status);
        status=read(portDescriptor, &processBuffer,50);
        printf("\nRead status for %s: %d\n",destinationBuffer, status);
    }

    processBuffer="\n";
    strcat(parameterValue[messagelIndex],processBuffer);
    status=write(portDescriptor, parameterValue[messagelIndex],
strlen(parameterValue[messagelIndex]));

    sleep(1);
    if(interactiveMode)
    {
        printf("Write status for %s: %d\n",
parameterValue[messagelIndex], status);
        status=read(portDescriptor, &processBuffer,50);
        printf("\nRead status for %s:
%d\n",parameterValue[messagelIndex], status);
    }

    decision=0x1A;
    status=write(portDescriptor, &decision ,1);

```

```

sleep(1);
if(interactiveMode)
{
    printf("Write status for 0x1A: %d\n", status);
    status=read(portDescriptor, &processBuffer,50);
    printf("\nRead status for 0x1A: %d\n",status);
}

portInit(0,whichPort);
return 0;

}

```

B.4.3. zaman.c

```

#include <time.h>
#include <stdio.h>

#define SIZE 256

char * zaman (void)
{
    char buffer[SIZE];
    time_t curtime;
    struct tm *loctime;

    /* Get the current time. */
    curtime = time (NULL);

    /* Convert it to local time representation. */
    loctime = localtime (&curtime);

```



```

/* Print it out in a nice format. */
strftime (buffer, SIZE, "[%d-%m-%Y %H:%M:%S]", loctime);
return buffer;
}

```

B.4.4. sms_server.c

```

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h> /* close */

#define SUCCESS 0
#define ERROR 1

#define END_LINE 0x0
#define SERVER_PORT 1500
#define MAX_MSG 160

/* function readline */
int read_line();
int sms_main(int parameterCount, char * parameterValue[], int output);
char * zaman (void);

int main (int argc, char *argv[]) {

    int sd, newSd, cliLen, pid, kontrol;

    int parc,i,output=1;

```

```

char *parv[20];
struct sockaddr_in cliAddr, servAddr;
char line[MAX_MSG];
char mesaj[MAX_MSG];
char telefon[MAX_MSG];
char temp[MAX_MSG][MAX_MSG];
FILE *f;

/* create socket */
sd = socket(AF_INET, SOCK_STREAM, 0);
if(sd<0) {
    perror("cannot open socket ");
    return ERROR;
}

/* bind server port */
servAddr.sin_family = AF_INET;
servAddr.sin_addr.s_addr = htonl(INADDR_ANY);
servAddr.sin_port = htons(SERVER_PORT);

if(bind(sd, (struct sockaddr *) &servAddr, sizeof(servAddr))<0) {
    perror("cannot bind port ");
    return ERROR;
}

listen(sd,5);

if(argc>1)
{
    if(!strcmp(argv[1],"-d"))
    {
        daemon(1,1);
        pid=getpid();

```

```

//printf("pid: %d\n\n",pid);
output=0;

if(f=fopen("/var/run/sms_server.pid","w"))
{
    fprintf(f,"%d",pid);
    fclose(f);
}
else
{
    printf("WARNING: failed to create /var/run/sms_server.pid\n");
}
}
}
else
{
    printf("/-----\\n");
    printf("|          Welcome to SMS server v1.0          |\n");
    printf("|-----|\n");
    printf("|SMS Server by Caglar ULKUDERNER\n");
    <caglar@ulkuderner.net>|\n");
    printf("|          |\n");
    printf("|          use -d for daemon mode          |\n");
    printf("\\-----/\n\n");
}
if(!output)
{
    if(f=fopen("/var/log/sms_server","a+"))
    {
        fprintf(f,"%s waiting for data on port TCP\n",zaman(),SERVER_PORT);
        fclose(f);
    }
}

```

```

else
{
    printf("WARNING fail to open /var/log/sms_server\n");
}
}

while(1) {
    if(output)
    {
        printf("%s waiting for data on port TCP %u\n",argv[0],SERVER_PORT);
    }
    cliLen = sizeof(cliAddr);
    newSd = accept(sd, (struct sockaddr *) &cliAddr, &cliLen);
    if(newSd<0) {
        perror("cannot accept connection ");
        return ERROR;
    }

    /* init line */
    memset(line,0x0,MAX_MSG);

    /* receive segments */
    parc=1;
    kontrol=0;

    while(read_line(newSd,line)!=ERROR) {

        if(output)
        {
            printf("%s: received from %s:TCP%d : %s\n", argv[0],
                inet_ntoa(cliAddr.sin_addr),
                ntohs(cliAddr.sin_port), line);
        }
    }
}

```

```

        strcpy(temp[kontrol],line);
        parv[parc]=temp[kontrol];
        kontrol++;
        parc++;

/* init line */
memset(line,0x0,MAX_MSG);

} /* while(read_line) */
if(!output)
{
    if(f=fopen("/var/log/sms_server","a+"))
    {
        fprintf(f,"%s received from %s : ",
zaman(),inet_ntoa(cliAddr.sin_addr));
        for(i=1;i<parc;i++)
        {
            fprintf(f,"%s ", parv[i]);
        }
        fprintf(f,"\n");
        fclose(f);
    }
}

sms_main(parc,parv,output);
if(!output)
{
    if(f=fopen("/var/log/sms_server","a+"))
    {
        fprintf(f,"%s Sending SMS-Core done. \n", zaman());
        fclose(f);
    }
}

```

```
    } /* while (1) */  
}
```

```
int read_line(int newSd, char *line_to_return) {
```

```
    static int rcv_ptr=0;
```

```
    static char rcv_msg[MAX_MSG];
```

```
    static int n;
```

```
    int offset;
```

```
    offset=0;
```

```
    while(1) {
```

```
        if(rcv_ptr==0) {
```

```
            /* read data from socket */
```

```
            memset(rcv_msg,0x0,MAX_MSG); /* init buffer */
```

```
            n = recv(newSd, rcv_msg, MAX_MSG, 0); /* wait for data */
```

```
            if (n<0) {
```

```
                perror(" cannot receive data ");
```

```
                return ERROR;
```

```
            } else if (n==0) {
```

```
                //printf(" connection closed by client\n");
```

```
                close(newSd);
```

```
                return ERROR;
```

```
            }
```

```
        }
```

```
        /* if new data read on socket */
```

```
        /* OR */
```

```
        /* if another line is still in buffer */
```

```
        /* copy line into 'line_to_return' */
```

```

while(*(rcv_msg+rcv_ptr)!=END_LINE && rcv_ptr<n) {
    memcpy(line_to_return+offset,rcv_msg+rcv_ptr,1);
    offset++;
    rcv_ptr++;
}

/* end of line + end of buffer => return line */
if(rcv_ptr==n-1) {
    /* set last byte to END_LINE */
    *(line_to_return+offset)=END_LINE;
    rcv_ptr=0;
    return ++offset;
}

/* end of line but still some data in buffer => return line */
if(rcv_ptr <n-1) {
    /* set last byte to END_LINE */
    *(line_to_return+offset)=END_LINE;
    rcv_ptr++;
    return ++offset;
}

/* end of buffer but line is not ended => */
/* wait for more data to arrive on socket */
if(rcv_ptr == n) {
    rcv_ptr = 0;
}

} /* while */
}

```

B.4.5. sms_client.c

```

/*
smsClient by Caglar Ulkuderner

```

```

*/

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h> /* close */

#define SERVER_PORT 1500
#define MAX_MSG 100

int main (int argc, char *argv[]) {

    int sd, rc, i;
    struct sockaddr_in localAddr, servAddr;
    struct hostent *h;

    if(argc < 3) {
        printf("SMS Client v1.0\n");
        printf("=====\n");
        printf("usage: %s <sms-gw_ip> <options>\n",argv[0]);
        printf("\nOptions:\n");
        printf("-c --center [number]:Defines an SMS center to work with
<Optional>\n");
        printf("-d --destination [number]: Defines the destination of the
message.\n");
        printf("-m --message [\"message\"]: The message itself.\n");
        printf("-p --port [n]: defines which serial port to be used. Use port
number-1 for n. (i.e. 0 for com port 1).\n");
        printf("\nSMS Client v1.0\nDeveloped by: Caglar Ulkuderner\nSpecial
thanks to Hakan BAYINDIR for SMS_CORE\n");
    }
}

```



```

        exit(1);
    }

    h = gethostbyname(argv[1]);
    if(h==NULL) {
        printf("%s: unknown host '%s'\n",argv[0],argv[1]);
        exit(1);
    }

    servAddr.sin_family = h->h_addrtype;
    memcpy((char *) &servAddr.sin_addr.s_addr, h->h_addr_list[0], h-
>h_length);
    servAddr.sin_port = htons(SERVER_PORT);

    /* create socket */
    sd = socket(AF_INET, SOCK_STREAM, 0);
    if(sd<0) {
        perror("cannot open socket ");
        exit(1);
    }

    /* bind any port number */
    localAddr.sin_family = AF_INET;
    localAddr.sin_addr.s_addr = htonl(INADDR_ANY);
    localAddr.sin_port = htons(0);

    rc = bind(sd, (struct sockaddr *) &localAddr, sizeof(localAddr));
    if(rc<0) {
        printf("%s: cannot bind port TCP %u\n",argv[0],SERVER_PORT);
        perror("error ");
        exit(1);
    }

```

```

/* connect to server */
rc = connect(sd, (struct sockaddr *) &servAddr, sizeof(servAddr));
if(rc<0) {
    perror("cannot connect ");
    exit(1);
}

for(i=2;i<argc;i++) {

    rc = send(sd, argv[i], strlen(argv[i]) + 1, 0);

    if(rc<0) {
        perror("cannot send data ");
        close(sd);
        exit(1);
    }

    //printf("%s: data%u sent (%s)\n",argv[0],i-1,argv[i]);

}
//printf("data sent to %s\n",argv[1]);
return 0;

}

```

B.4.6. Makefile

```

CFLAGS=-g -WALL
LDFLAGS= -s
all:
    @echo "For SMS_Server: make server"
    @echo "For SMS_Client: make client"

```

```
@echo "For Clean : make clean"
server: sms_server.c messageSendingCore.c corePortAccess.c zaman.c
gcc -g -c -Wall sms_server.c -o sms_server.o
gcc -g -c -Wall messageSendingCore.c -o messageSendingCore.o
gcc -g -c -Wall corePortAccess.c -o corePortAccess.o
gcc -g -c -Wall zaman.c -o zaman.o
gcc sms_server.o messageSendingCore.o corePortAccess.o
zaman.o -o sms_server
client: sms_client.c
gcc sms_client.c -o sms_client

clean:
rm -rf zaman.o corePortAccess.o messageSendingCore.o
sms_server.o sms_server sms_client
```

B.4.7. SMSd

```
#!/bin/bash
#
# SMS Server control script
# Caglar Ulkuderner

# source function library
. /etc/rc.d/init.d/functions

# Variables
RETVAL=0
prog="SMSd"
PID_FILE=/var/run/sms_server.pid

start()
```

```

{
    CURDIR=`pwd`
    echo -n $"Starting $prog:"
    /usr/bin/sms_server -d
    echo_success
    RETVAL=$?
    echo
}

stop()
{
    echo -n $"Stopping $prog:"
    pid=`cat $PID_FILE`
    kill -9 $pid
    rm -rf $PID_FILE
    echo_success
    RETVAL=$?
    echo
}

status()
{
    if `test -f $PID_FILE`
    then
        echo "PID(`cat $PID_FILE`) Running.."
    else
        echo "Stoped.."
    fi
}

case "$1" in
    start)
        start

```

```
        ;;
stop)
    stop
    ;;
restart)
    stop
    sleep 5
    start
    ;;
status)
    status
    ;;
*)
    echo $"Usage: $0 {start|stop|restart|status}"
    RETVAL=1
esac
exit $RETVAL
```