Bastian Ballmann

Understanding Network Hacks

Attack and Defense with Python 3

2nd Edition

Springer

Understanding Network Hacks

Bastian Ballmann

Understanding Network Hacks

Attack and Defense with Python 3

2nd Edition

Springer

Bastian Ballmann
Uster, Switzerland

ISBN 978-3-662-62156-1 ISBN 978-3-662-62157-8 (eBook)
https://doi.org/10.1007/978-3-662-62157-8

Responsible Editor: Martin Börger
This Springer imprint is published by the registered company Springer-Verlag GmbH, DE part of Springer Nature.
The registered company address is: Heidelberger Platz 3, 14197 Berlin, Germany

For data travelers, knowledge hungry, curious, network-loving lifeforms who like to explore and get to the bottom of thing.

Foreword

Doesn't this book explain how to break into a computer system? Isn't that illegal and a bad thing at all?

I would like to answer both questions with no (at least the second one). Knowledge is never illegal nor something bad, but the things you do with it.

You as an admin, programmer, IT manager or just an interested reader cannot protect yourself if you don't know the techniques of the attackers. You cannot test the effectiveness of your firewalls and intrusion detection systems or other security related software if you are not able to see your IT infrastructure through the eyes of an attacker. You cannot weigh up the danger to costs of possible security solutions if you don't know the risks of a successful attack. Therefore it is necessary to understand how attacks on computer networks really work.

The book presents a selection of possible attacks with short source code samples to demonstrate how easy and effectively and maybe undetected a network can be infiltrated. This way you can not only learn the real techniques, but present them to your manager or employer and help them in the decision if it would make sense to care a little bit more about IT security. At the end of the book you should be able to not only understand how attacks on computer networks really work, but also to modify the examples to your own environment and your own needs.

Sure, the book also tells those bad guys how to crack the net and write their own tools, but IT security is a sword with two sharp blades. Both sides feed themselves off the same pot of knowledge and it is an continuous battle, which the protecting side can never dream of winning if it censors itself or criminalizes their knowledge!

Introduction

Who should Read this Book?
This book addresses interested Python programmers who want to learn about network coding and to administrators, who want to actively check the security of their systems and networks. The content should also be useful for white, gray and black hat hackers, who prefer Python for coding, as well as for curious computer users, who want to get their hands on practical IT security and are interested in learning to see their network through the eyes of an attacker.

You neither need deep knowledge on how computer networks are build up nor in programming. You will get throught all the knowledge you need to understand the source codes of the book in Chaps. 2 and 3. Readers, who know how to program in Python and dream in OSI layers or packets headers can right away jump to Chap. 5 and start having fun at their device.

Of course a book like this needs a disclaimer and the author would be happy if all readers only play on systems they are allowed to do so and use the information of this book only for good and ethical actions otherwise you maybe breaking a law depending on the country your device is connected in.

The length of the book doesn't allow for in depth discussion of all topics. You will only get somewhat more than the basics. If you want to dig deeper you should afterwards get some special lecture in your special field of interest.

The Structure of the Book

The different hacks are grouped by network protocols and every chapters content is ordered by difficulty. You can read the book in the order you like except the both introduction chapters about networks (Chap. 2) and Python (Chap. 3).

The code samples are printed unshortened therefore you can just copy and use them without worrying about incremental changes or addons. All source codes presented in this book can also be found on Github at https://github.com/balle/python-network-hacks.

At the end of each chapter you will find a selection of tools also written in Python that attack the described protocol in a more detailed way.

Thanks to the basic knowledge learned in the chapter it shouldn't be too hard to read and understand the source code of the tools.

The Most Important Security Principles

The most important principles in building a secure network of the authors point of view are:

1. Security solutions should be simple. A firewall rule-set that no one understands, is a guarantee for security holes. Software that's complex has more bugs than simple code.
2. Less is more. More code, more systems, more services provide more possibilities of attack.
3. Security solutions should be Open Source. You can easier search for security problems if you have access to the source code. If the vendor disagrees to close an important security hole you or someone else can fix it and you don't have to wait for six or more months till the next patch day. Proprietary software can have build in backdoors sometimes called Law Interception Interface. Companies like Cisco (see RFC 3924), Skype (US-Patent-No 20110153809) and Microsoft (e.g. _NSAKEY http://en.wikipedia.org/wiki/NSAKEY) are only popular examples.
4. A firewall is a concept not a box that you plug in and you are safe.
5. Keep all your systems up to date! A system that's considered secure today can be unprotected a few hours later. Update all systems, also smart phones, printer and switches!
6. The weakest device defines the security of the complete system and that doesn't necessarily have to be a computer it can also be a human (read about social engineering).
7. There is no such thing as 100% secure. Even a computer that is switched off can be infiltrated by a good social engineer. The aim should be to build that much layers that the attacker falls over one tripwire and leaves traces and that the value he or she can gain from a successful infiltration is much lower than the effort to attack or that it exceeds the intruders skills.

Contents

1 Installation

Abstract

This chapter explains on which operating system the sources can be executed, which Python version you will need and how to install additional Python modules. Last but not least, we will discuss some possible solutions for setting up a complete development environment. If you are already familiar with the Python programming language you can skip this introductory chapter without missing anything.

1.1 The Right Operating System

Yes, I know the title of this section can lead to flame wars. It should just illustrate on which operating systems the source codes of this book are run. The author is using a GNU/Linux systems with kernel version 5.x for development, but most of the sources, except the chapter about Bluetooth, should also runable on BSD or Mac OS X systems. If you succeed in running the source code on other systems the author would be happy if you could drop him a tiny email. Of course all other comments or criticisms are also welcome.

1.2 The Right Python Version

All source code examples are written in Python 3 and have been tested with Python 3.7.

To check which version of Python is installed on your system, execute the following command

```
python3 --version
Python 3.7.4
```

B. Ballmann, *Understanding Network Hacks*,
https://doi.org/10.1007/978-3-662-62157-8_1

1.3 Development Environment

The author prefers GNU/Emacs (www.gnu.org/software/emacs) as a development environment, because he thinks its editing and extension possibilities are unbeatable. Emacs supports all common features like syntax highlighting, code completion, code templates, debugger support, PyLint integration and thanks to Rope, Pymacs and Ropemacs, it has one of the best refactoring support for Python.

If you want to give Emacs and it features a try, the author suggests installing the awesome extension set Emacs-for-Python, downloadable at gabrielelanaro.github.com/emacs-for-python. Thanks to the amount of available plugins, Emacs can also be used as an email and Usenet client, for irc or jabber chatting, as music player and additional features like speech support, integrated shell and file explorer up to games like Tetris and Go. Some guys even think Emacs is not an IDE, but a whole operating system and use it as init process.

A good alternative for a console editor is Vim (www.vim.org/) of course. The author does not like flame wars so if you do not know Emacs or Vim, give both a try. They are great! Vim includes all features of a modern IDE, is extensible and completely controllable with keyboard shortcuts and features a GUI version.

If you want to use one of those full-blown, modern IDEs, then check out Eclipse (www.eclipse.org/) together with PyDev (pydev.org/). Eclipse also has all the common features as well as code outlining, a better integrated debugging support and an endless seeming torrent of useful plugins like UMLet to draw UML diagrams or Mylyn to perfectly integrate a bugtracking system.

As alternative GUI-only IDE, you could also check out Eric4 (eric-ide.python-projects.org/) and Spyder (code.g.oogle.com/p/spyderlib/), which also include all common features plus a debugger, PyLint support and refactoring.

If you do not have that many resources and RAM for programming tasks, but need a GUI then Gedit might be the editor of your choice. However you should extend it with a bunch of plugins: Class Browser, External Tools, PyLint, Python Code Completion, Python Doc String Wizard, Python Outline, Source Code Comments and Rope Plugin.

The installation could be somewhat nasty and the functionality not as complete as for the other candidates. However, Gedit only uses the tenth of your RAM that Eclipse does.

The final choice is left to you. If you don't want to choose or try all possibilities, you should first try Eclipse with Pydev as bundle downloadable from Aptana (aptana.com/products/studio3). The chances are high that you will like it.

1.4 Python Modules

Python modules can be found in the Python packet index pypi.python.org. New modules can be installed by one of the following three possibilities:

1 Download the source archive, unpack it and execute the magic line

```
python3 setup.py install
```

2 Use easy_install

```
easy_install <modulname>
```

3 Get your feet wet with pip. Maybe you have to install a package like `python-pip` before you can use it.

```
pip3 install <modulname>
```

You should use `pip`, because it also supports deinstallation and upgrading of one or all modules. You could also export a list of installed modules and its version, reinstall them on another system, you can search for modules and more.

Alternatively you can tell pip to install the modules in a directory of your homedir by adding the parameter `–user`.

Which Python modules are needed for which tools and source code snippets will be described at the beginning of the chapter or in the description of the snippet, if the module is only used for that code. This way, you will only install modules that you really want to use.

1.5 Pip

With Pip you can also search for a module.

```
pip search <modulname>
```

To uninstall a module just use the option `uninstall`. A listing of all installed modules and their versions can be achieved with the parameter `freeze` and later on used to reinstall them.

```
pip3 freeze > requirements.txt
pip3 install -r requirements.txt
```

Which modules are outdated reveas the command `pip list –outdated`. A single module can be upgraded by executing `pip3 install –upgrade <modulname>`.

1.6 Virtualenv

If you like you could install all Python modules needed for this book in a subfolder (a so called virtualenv) so that they wont conflict with the modules installed in your operating system. As an example we will create a virtualenv called `python-network-hacks`, install the module `scapy` into it and exit from the virtual environment.

```
python3 -m venv python-network-hacks
source python-network-hacks/bin/activate
(python-network-hacks) $ pip3 install scapy
(python-network-hacks) $ deactivate
$ _
```

Make sure that the prompt is the default prompt again after deactivating.

Network 4 Newbies

2

Abstract

Computer networks are the veins of the information age, protocols the language of the net. This chapter describes the basics of networking starting with hardware going over to topology and the functionality of the most common protocols of an Ethernet/IP/TCP network up to Man-in-the-middle attacks. For all who want to rebuild or refresh their knowledge of networking.

2.1 Components

To be able to build a computer network of course you need some hardware. Depending on the kind of net you'll need cables, modems, old school acoustic in banana boxes, antennas or satellite receivers beside computers and network cards as well as router (Sect. 2.14), gateways (Sect. 2.13), firewalls (Sect. 2.18), bridges (Sect. 2.15), hubs and switches.

A **hub** is just a simple box you plug network cables in and it will copy all signals to all connected ports. This property will probably lead to an explosion of network traffic. That's a reason why hubs are rarely used these days. Instead most of the time you will see **switches** building the heart of the network. The difference between a hub and a switch is a switch remembers the MAC address of the network card connected to the port and sends traffic only to the port it's destinated to. MAC addresses will be explained in more detail in Sect. 2.4.

B. Ballmann, *Understanding Network Hacks*,
https://doi.org/10.1007/978-3-662-62157-8_2

2.2 Topologies

You can cable and construct computer networks in different ways. Nowadays the most common variant is the so called **star network** (see Fig. 2.1), where all computer are connected to a central device. The disadvantage is that this device is a single point of failure and the whole network will break down if it gets lost. This disadvantage can be circumstanced by using redundant (multiple) devices.

Another possibility is to connect all computers in one long row one after the other, the so called **bus network** (see Fig. 2.2). The disadvantage of this topology is that each computer must have two network cards and depending on the destination the traffic gets routed through all computers of the net. If one of them fails or has too high a load the connections behind that host are lost.

The author has seen only a few bus networks this decade and all consisted of two computers directly connected to guarantee time critical or traffic intensive services like database replication, clustering of application servers or synchronization of backup servers. In all cases the reason for a bus network was to lower the load of the star network.

As last variant the **ring network** (Fig. 2.3) should be mentioned, which as the name implies connects all computers in a circle. The ring network has the same disadvantages as a bus network except that the network will only fail partly if a computer gets lost as long as the net can route the traffic the other way round. The author has not seen a productive ring network, but some wise guys whisper that it it the topology of backbones used by ISPs and large companies.

Additionally one often reads about **LAN** (Local Area Network), **WAN** (Wide Area Network) and sometimes even about **MAN** (Middle Area Network). A LAN is a local network that's most of the time limited to a building, floor or room.

In modern networks most computers are connected on a LAN over one or more switches. Multiple LANs connected over a router or VPN (see Sect. 2.17) are called MAN. If the

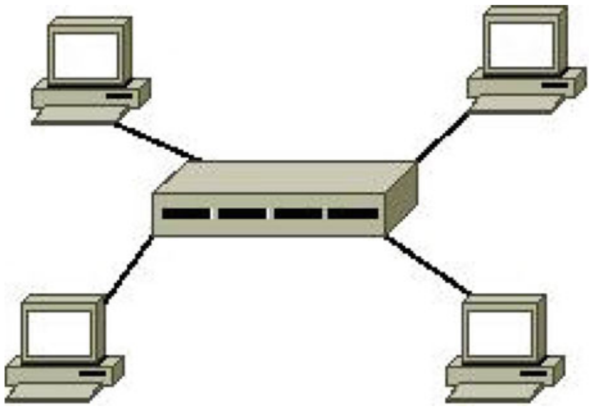


Fig. 2.1 star network

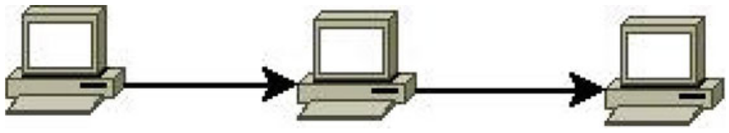
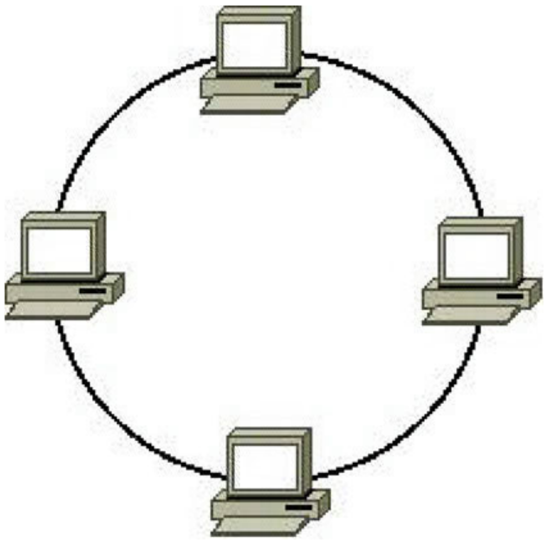


Fig. 2.2 Bus network

Fig. 2.3 Ring network



network spreads over multiple countries or even the whole world like the internet than it is defined as a WAN.

2.3 ISO/OSI Layer Model

According to the pure doctrine the ISO/OSI layer model, technically separates a computer network into seven layers (see Fig. 2.4).

Each layer has a clearly defined task and each packet passes them one after another in the operating systems kernel up to the layer it's operating on (Table 2.1).

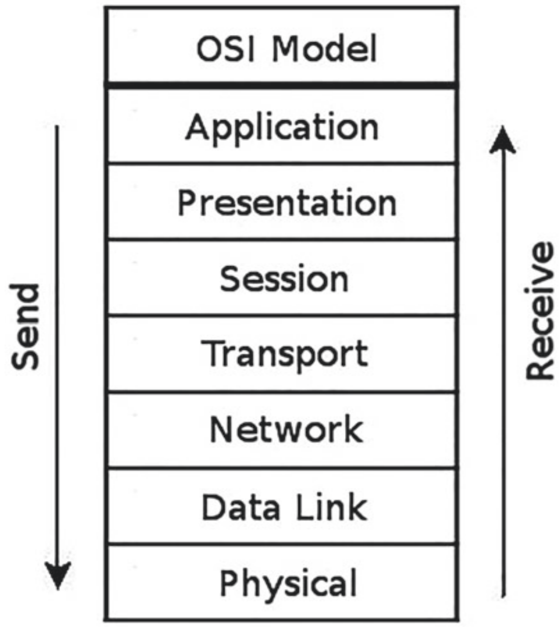


Fig. 2.4 OSI model

Table 2.1 OSI layer

OSI layer	Layer name	Task
1	Physical	Cables, Antennas, etc.
2	Data-Link	Creates a point-to-point connection between two computers
3	Network	Provides for addressing of the destination system
4	Transport	Takes care that the data is received in the right order and enables retransmission on packet loss
5	Session	Used to address single applications (e.g. using ports)
6	Presentation	Conversion of data formats (e.g. byte order, compression, encryption)
7	Application	Protocols that define the real service like HTTP

2.4 Ethernet

Have you ever bought a "normal" network cable or card in a shop? Than the chance is nearly 100% that you own ethernet hardware, because Ethernet is with huge margin the most used network technology today. You will see network components with different speed limits like 1, 10, 100 MBit or gigabit and an ethernet can be constructed with different cable types like coaxial (old school), twisted pair (common) or glass fiber (for data hungry guys).

Twisted pair cables can be divided into to the variations **STP** (Single Twisted Pair) and **UTP** (Unshielded Twisted Pair) as well as patch- and crossover cables.

The difference between STP and UTP cables is that the fibers of the UTP cables are unshielded and therefore they have a lower quality compared to STP cables. Nowadays new cables in a shop should all be STP.

Patch and cross cables can be separated from each other by looking at the plugs of the cable. If the colors of the fibers are in the same order than its a patch otherwise a cross cable. A **cross cable** is used to directly connect two computers, a **patch cable** is used to connect a computer to a hub or switch. Modern network cards can automatically cross the fibers so cross cables are a dying race.

Every network card in an Ethernet network has a MAC address that's worldwide unique and are used to address devices on the net. The **MAC address** consists of six two digit hexadecimal numbers, which are separated by colons (e.g. `aa:bb:cc:11:22:33`).

Its a common misbelief that a computer in a local TCP/IP network is reached over its IP address; in reality the MAC address is used for this purpose. Another common misunderstanding is that the MAC address cannot be spoofed. The operating system is responsible to write the MAC into the Ethernet header and systems like GNU/Linux or *BSD have possibilities in their base system to change the MAC with one command.

```
ifconfig enp3s0f1 hw ether c0:de:de:ad:be:ef
```

Beside the source destination MAC address an Ethernet header (see Fig. 2.5) consists of a type field and a checksum. The type field defines the protocol that follows Ethernet e.g. 0x0800 for IP or 0x0806 for ARP.

Last but not least the term CSMA/CD should be explained. CSMA/CD stands for Carrier Sense Multiple Access/Collision Detect and describes how a computer sends data over an Ethernet. First of all it listens on the wire if someone is currently sending something. If that's the case it just waits a couple of random seconds and tries again. If the channel is free it sends the data over the network. Should two stations be transmitting data at the same data a collusion will result, therefore every sending station must listen afterwards to detect a collusion, than randomly wait some seconds and retransmit the data.

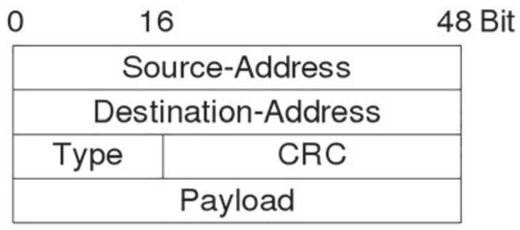


Fig. 2.5 Ethernet header

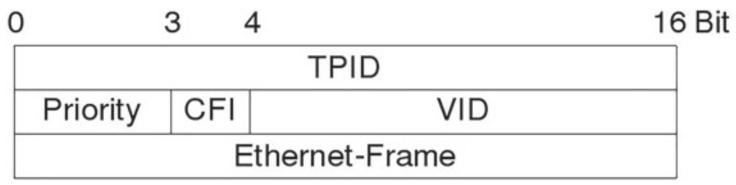


Fig. 2.6 VLAN header

2.5 VLAN

A VLAN (Virtual Local Area Network) separates several networks on a logical base. Only devices on the same VLAN can see each other. VLANs where invented to define a networks structure independently from its physical hardware, to prioritize connections and to minimize broadcast traffic. They were not developed with security in mind, but its a common myth that VLANs can add to your security. Don't rely on this myth, because several ways exist to circumvent the separation of a VLAN (see Sect. 4.5).

Switches implement VLANs in two different ways: through tagging of packets using a IEEE 802.1q Header (see Fig. 2.6), that's inserted after the Ethernet header or simply defined by port. 802.1q is a newer variant, which allows the creation of a VLAN spread over several switches.

2.6 ARP

ARP (Address Resolution Protocol) translates between layer 2 (Ethernet) and 3 (IP). It is used to resolve MAC addresses to IP addresses. The other way round is done by RARP (Reverse Address Resolution Protocol). The structure of an ARP headers can be seen in Fig. 2.7.

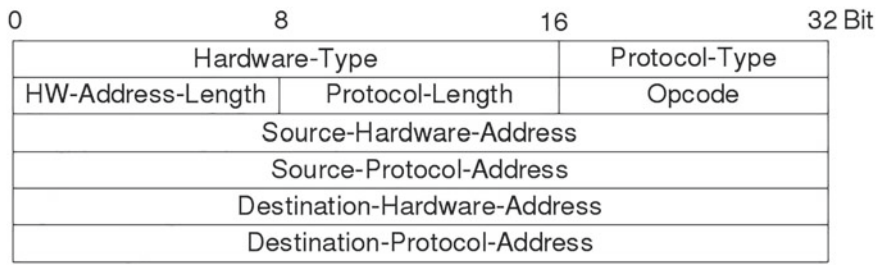


Fig. 2.7 ARP header