OS Hardening

Seminararbeit

Ausgewählte Kapitel der IT-Security

Vorgelegt von:
Nicolas Schulterer

Personenkennzeichen
2110475125

Abgabe am:
20.12.2023

Glossary

OS	Operating System
IoT	Internet of Things
OWASP	Open Web Application Security Project
NIST	National Institute of Standards and Technology
CIS	Center for Internet Security

Keywords

OS
Hardening
Linux
Exploits
Vulnerabilities

Contents

Chapter 1

Introduction

In today's interconnected world, the quest for absolute security in computer systems remains an elusive goal. Operating systems (OS) are tasked with the complex challenge of managing multiple programs and applications simultaneously, some of which originate from third party sources beyond the OS's control. It's crucial for both users and administrators to acknowledge this inherent complexity when contemplating the safeguarding of their systems against external and internal threats. One potent strategy in this ongoing battle for security is known as system hardening.

System hardening is "locking a system down and reducing its attack surface" entails a deliberate effort to eliminate any unnecessary programs or applications, changing default configurations to their most secure settings and only execute essential processes. [Cog23] These principles are equally applicable when hardening any OS.

In this seminar paper, we delve into the realm of OS hardening, with a primary focus on the Linux operating system—a widely adopted and well-established OS in the computing world. We will begin by exploring the concept of OS hardening in more detail, highlighting the risks associated with neglecting this critical practice. Additionally, we will examine the prevailing threats that pose challenges to the security of Linux systems. Finally, we will embark on a journey to harden a vulnerable Linux OS, demonstrating several tools that are useful for Hardening Linux.

Chapter 2

Background

In light of the evolving threat landscape, the importance of system hardening has never been more evident. Recent developments, such as the allocation of a record-breaking $12.7 billion in the fiscal year 2024 US Government budget for civilian sector cybersecurity [Hou23], underscore the critical nature of robust security measures. This significant investment by the US government serves as a clear indicator of the potential consequences that weak security infrastructure can impose on a nation.

However, the importance of system hardening extends beyond the realm of governments and national security. Businesses and private users should take heed of these statistics, recognizing the potential damage that inadequate system security can inflict. This awareness becomes paramount when considering the storage of sensitive information on an OS.

To navigate the intricacies of OS hardening effectively, one must first grasp the fundamentals: What does it mean to harden a system? What are the risks associated with neglecting system hardening? And what are the specific threats that system hardening can mitigate, especially within the Linux OS?

2.1 Understanding OS Hardening

In our pursuit of system security, OS hardening stands as a crucial pillar. But what exactly does it entail? As mentioned previously, OS hardening comprises a comprehensive set of precautions that users and administrators employ to bolster the security of their systems. Its primary objective is to reduce the potential attack surface available to malicious actors. This is usually achieved by disabling or removing any unnecessary application or software, configuring default configurations to their most secure state, execute only the processes that are explicitly required, securing unused permissions, ports, user accounts, amongst others. Having a "anything that is not specifically allowed is restricted" mentality when making security decisions.[Cog23, Dan23, Jog17, NMZL14]

To embark on the journey of hardening an OS effectively, it is advisable to follow established guidelines, provided by several respected organizations such as the National Institute of Standards and Technology (NIST) [oST17], the Center for Internet Security (CIS) [fIS], OWASP [OWA], and the Australian Cyber Security Centre (ACSC)

[Cen23], offer valuable resources in this regard. These guidelines serve as practical road maps to help users and administrators secure their OS with best practices in mind.

2.2 The Consequences of Insufficient OS Hardening

As highlighted in the previous section, OS hardening is a crucial practice. But why do we invest time and effort into hardening our systems? The answer lies in the significant risks posed by insufficiently secured operating systems, risks that affect not only governments and businesses but also everyday users who value the security of their sensitive information. The gravest consequence of inadequate system security is the occurrence of data breaches. In these scenarios, private personal information for individual users may be compromised, potentially impacting their daily lives. For businesses, such breaches can result in severe repercussions, including financial losses, regulatory fines, and damage to their reputation within their respective industries. [Cog23] For governments, the consequences can extend far beyond trust and reputation damage, potentially jeopardizing the safety and well-being of their citizens, as witnessed in the July 2023 US government hack. [SS23]

These breaches are often attributable to human error, which can stem from security vulnerabilities within the operating system or associated software. Such vulnerabilities could have been mitigated if users or administrators had regularly applied security updates and patches. One contributing factor to these errors is the lack of proper training for users or administrators. [Tev20]

To stay informed about the ever expanding landscape of risks, it is advisable to follow security news from reliable sources like Hacker News, Ars Technica, and Fudzilla amongst others. Staying informed is a proactive step towards understanding and mitigating the potential threats to your system. [Tev20]

2.3 Threats to Linux

Having discussed the inherent risks associated with neglecting OS hardening, this section narrows its focus to examine the specific threats that target the Linux OS. Linux, as an open source and widely used OS, faces a unique set of challenges and vulnerabilities. In this paper, we delve into these Linux specific threats to gain a deeper understanding of the security landscape in the realm of Linux.

The foremost challenge confronting Linux is its insecure default settings. This vulnerability can lead to a myriad of security headaches. Furthermore, with the ever growing proliferation of Internet of Things (IoT) devices powered by Linux as their OS, this issue becomes increasingly critical. Regrettably, there have been numerous instances of security mishaps and vulnerabilities arising from the inadequate configuration of these devices, often stemming from the lack of security awareness among users and administrators. [Tev20]

Linux, although not prone to traditional virus infections, remains a target for a wide range of malware and attacks. Among these, Botnet malware, Ransomware,

exemplified by the infamous WannaCry incident which gained notoriety, largely due to extensive media coverage [MTFM17] , and Cryptocurrency mining software have emerged as favored tools for attackers seeking to compromise Linux systems. However, not all threats are malware-based; attackers have devised alternative methods for extracting sensitive personal information, including credentials and credit card details. [Tev20] It's not like the amount of ways to breach a system is decreasing either, like freshly discovered Looney Tunables (CVE-2023-4911) vulnerability [New23], which will be mentioned in more detailed below.

When discussing specific threats and vulnerabilities exploited by attackers to compromise a system, we must begin with one of the most widely recognized issues in IT security: weak passwords. Weak passwords, particularly when assigned to the root account, represent an easily exploitable avenue for attackers to gain control over a system, potentially using it as part of a botnet. [Tev20]

The section below lists common vulnerabilities of Linux systems which helps understand their threat to potential devices:

- **Privilege Escalation** is an exploitation technique that grants attackers elevated access to resources typically protected from regular users or programs. This enables them to perform actions otherwise restricted to administrators or software developers. [NMZL14] This exploit poses a significant threat, with a track record of effectiveness and a growing popularity in recent years. As previously mentioned, 'Loony Tunables' represents the latest addition to an ever-expanding list of privilege escalation vulnerabilities discovered in recent years. Notable examples include 'Baron Samedit' (CVE-2021-3156), 'Sequoia' (CVE-2021-33909), and 'PwnKit' (CVE-2021-4034). [New23]

- **Denial of Service (DoS)** is an exploitation technique that leverages flaws in network protocol implementations or exhausts a target system's resources through brute force. Its primary objective is to disrupt normal service provision or block access to resources, potentially causing the target to freeze or even crash. [NMZL14] Although Linux is inherently resistant to certain types of DoS attacks with its default settings, it remains vulnerable to others. Therefore, it is crucial to conduct thorough research to determine which types of DoS attacks necessitate hardening measures and where resource allocation is unnecessary. [Tev20]

- **IP Spoofing** typically entails an attacker attempting to transmit network packets with a falsified IP address to a system, thereby deceiving the recipient. [Tev20] This deceptive practice involves forging the source address of a transmission with the aim of illicitly gaining entry into a secure system. [BLLB+23] IP Spoofing can be employed for various malicious activities, including launching a Denial of Service (DoS) attack, conducting anonymous port scanning, or deceiving access controls. [Tev20]

- **Apache Web Server vulnerabilities** pose an ongoing threat, inviting potential exploitation by attackers. The severity of their impact on a system varies, ranging from low for issues considered challenging to exploit or yielding minimal consequences, to critical instances where a remote attacker could exploit Apache

to execute arbitrary code. [Fou] A notable example is CVE-2014-6271, commonly known as Shellshock, affecting Apache2 Web Server. This vulnerability enables unauthorized access and the execution of arbitrary code on systems utilizing Bash, showcasing the importance of robust security measures in safeguarding against such exploits. [oST]

- **SSH security issues** are often overlooked compared to those of other technologies. This misconception stems from the belief that SSH is inherently secure by default; however, the default configuration of SSH is, in reality, vulnerable. [Tev20] Notably, SSH version 1 should *never* be employed due to its inherent security risks. Moreover, root user login, username/password logins, weak encryption algorithms, and default port settings are among several potential entry points for attackers attempting to breach your system. [Tev20]

- **Telnet security issues** are so pronounced that they hardly warrant extensive discussion. Not only is Telnet inherently insecure due to its lack of robust encryption for communication, but its use in a secure setting is *strongly discouraged*. Telnet should only be considered when absolutely necessary, and even then, it must be tunneled through an encrypted protocol to mitigate its inherent security vulnerabilities. [SJT08]

- **FTP** is an Internet standard for file transfer, facilitates the exchange of web pages, graphics, and other files between local and remote servers. [SPT+23] However, the historical design of FTP exposes inherent insecurities. Developed in an era with less emphasis on network security, FTP lacks crucial safeguards for file privacy, integrity, and user anonymity. Notably, all passwords and data are transmitted in plain text, making them susceptible to interception. Recognizing these vulnerabilities, the IETF addressed security concerns in an informational RFC document (RFC 2577). Additionally, FTP faces challenges in navigating modern network features like firewalls and address translation. [Aca23]

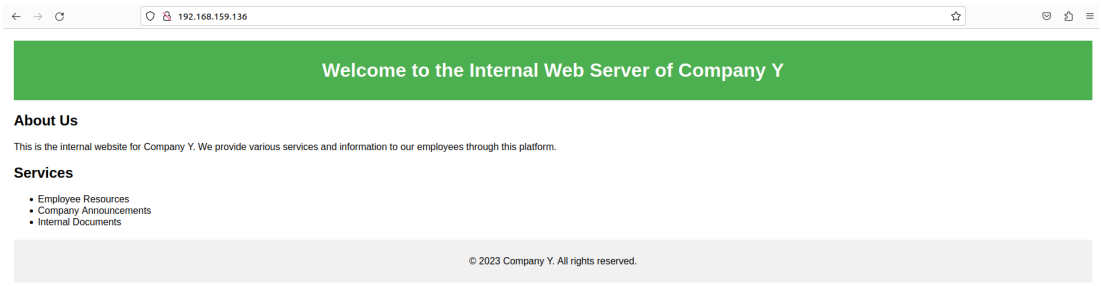
  Given FTP's susceptibility to eavesdropping, it is advisable for use only within trusted networks. In less secure environments, such as the public Internet, adopting secure alternatives like SFTP or SCP is imperative. SFTP not only matches FTP's file transfer capabilities but also enhances security by safeguarding user identities, login credentials, and transferred data. This approach effectively addresses and mitigates the inherent risks associated with FTP, particularly in Linux environments. [Aca23]

While there are numerous Linux vulnerabilities that may be exploited by malicious actors, an exhaustive list is beyond the scope of this paper. It is advisable for individuals to conduct thorough research tailored to their specific operating system, identifying potential threats and effective mitigation strategies. By staying informed about the evolving landscape of Linux security, users can proactively address vulnerabilities and prevent them from becoming significant issues.

Chapter 3

Linux OS Hardening

In this section, we will dive into the practical aspect of Linux OS hardening, demonstrating some hardening tools and explaining their purpose and usage. To illustrate the process, we will use an imaginary scenario involving 'Company Y', a business that has reached out for our expertise in securing one of their Ubuntu servers. 'Company Y' uses this server as an internal Web Server, that is accessible via Secure Shell (SSH). Our objective is to harden this server with the tools explained in detail below to ensure it's resilient against potential security threats. For this scenario we will running two virtual machines on "VMWare Workstation 17 Player", the web server being a server version of Ubuntu 22.04.03 running SSH and Apache2 Web Server, with the IP address 192.168.159.136, the inernal website can bee seen in Figure 3.1 and the second virtual machine, with the IP address 192.168.159.128, essentially being the "administrator" remotely accessing the server through SSH to harden the server.



Source: Author

Figure 3.1: Internal Website of Company Y

3.1 Hardening SSH

As detailed in Subchapter 2.3, the default configuration of SSH poses notable security risks, including vulnerabilities related to root user login, user/password logins, weak encryption, and default port settings. To address these potential threats, we will implement SSH hardening measures that go beyond its default configurations. Our strategy involves disabling SSH version 1, preventing root user login, changing the default port, configuring passwordless logins, and disabling username/password login.

To initiate this process, we will disable SSH version 1. The initial step is to connect to the server via SSH, as demonstrated in Figure 3.2

```
nico@nico-virtual-machine:~$ ssh company_y@192.168.159.136
The authenticity of host '192.168.159.136 (192.168.159.136)' can't be established.
ED25519 key fingerprint is SHA256:3rGkjyyOZ7I1tJB0u9ggXNYlJ2gNszSkkOEdXS09wSc.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.159.136' (ED25519) to the list of known hosts.
company_y@192.168.159.136's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-91-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Dec 20 10:14:31 AM UTC 2023

  System load:  0.0               Processes:             208
  Usage of /:   30.1% of 9.75GB   Users logged in:       1
  Memory usage: 9%                IPv4 address for ens33: 192.168.159.136
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

44 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Wed Dec 20 00:29:06 2023
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

company_y@web-server:~$
```

Source: Author

Figure 3.2: Connecting to the server via SSH

With remote access to the server established, our next step involves the hardening of SSH. To check if SSH version 1 is enabled, examine the configuration file `sshd_config`, typically located at `/etc/ssh/sshd_config`, Figure 3.3 shows how this file looks like. The presence of `Protocol 1` or `Protocol 1, 2` in the file, would indicate the activation of SSH version 1. However, if this line is not found, it signifies that version 1 is not enabled. In our case, this line is not present in the file.

The subsequent step is to disable root user login. In `/etc/ssh/sshd_config`, locate the line `PermitRootLogin`. On Ubuntu the default setting for SSH is `PermitRootLogin prohibit-password` allowing root users to log in through public key exchange. However, best practice recommends forcing admin users to use normal user accounts and use `sudo` for administrative actions. [Tev20] Uncomment this line and change it to `PermitRootLogin no`. Save and exit the file, then restart SSH with `sudo systemctl restart ssh`.

```
# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin prohibit-password
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile     .ssh/authorized_keys .ssh/authorized_keys2

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
KbdInteractiveAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
```

Source: Author

Figure 3.3: File /etc/ssh/sshd_config

Staying in the same section, changing the default port is also done in `/etc/ssh/sshd_config`, as depicted in Figure 3.3. The line `Port 22` sets the default port to 22. To enhance security, change it to a custom SSH port, for example, `Port 8022`, since it is not in the well-known range and is less likely to be scanned frequently. Save, exit, and restart SSH.

```
nico@nico-virtual-machine:~$ ssh-keygen -t rsa -b 3072
Generating public/private rsa key pair.
Enter file in which to save the key (/home/nico/.ssh/id_rsa): /home/nico/.ssh/key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/nico/.ssh/key
Your public key has been saved in /home/nico/.ssh/key.pub
The key fingerprint is:
SHA256:zske3qalVjzf4ICxvoQllKWxHMo+266lcgSTcsk32eA nico@nico-virtual-machine
The key's randomart image is:
+---[RSA 3072]----+
|      o .        |
|   ..o B         |
| . +o+*          |
|. B.E.. .        |
| o +o.. S=       |
|    .+ Bo.= .    |
|   .. +.B..= o   |
|  . .+ +o=. o .  |
|   oo...*+.      |
+----[SHA256]-----+
nico@nico-virtual-machine:~$ ls .ssh
key  key.pub  known_hosts  known_hosts.old
nico@nico-virtual-machine:~$
```

Source: Author

Figure 3.4: Creating a user's SSH key pair

Finally, set up passwordless login and disable username/password login. First we will create a SSH key pair on our administrator virtual machine. We will be using an RSA key of at least 3072 bits to meet the guidelines of NIST. [Tev20] To do this we will input the command `ssh-keygen -t rsa -b 3072` in the terminal, then the file where key will be saved to, in our case we will save it to `/home/nico/.ssh/key`, and then finally setting a passphrase for the key. We can make sure this key pair was generated by typing `ls .ssh` in the terminal. These steps can be seen in Figure 3.4 showing at the end both the private and public key.

In order for us to login without a password we have to transfer the public key to the remote server. These next step should still be done on the machine that will try to connect to the server. Before we can transfer the public key to the server we need to add the private key to the session keyring. This requires two commands: `exec /usr/bin/ssh-agent $SHELL`, which invokes ssh-agent and to add the private key to the keyring we require `ssh-add /home/user/.ssh/key` as seen in Figure 3.5.

```
nico@nico-virtual-machine:~$ ssh-add /home/nico/.ssh/key
Enter passphrase for /home/nico/.ssh/key:
Identity added: /home/nico/.ssh/key (nico@nico-virtual-machine)
```

Source: Author

Figure 3.5: Adding private key to keyring

The last step involves transferring the public key to the Web Server. This is done with `ssh-copy-id company_y@192.168.159.136` and then inserting the user's password, as seen in Figure 3.6. We can see with Figure 3.7 that now when we connect to the server via SSH it will use the key exchange and not a password.

```
nico@nico-virtual-machine:~$ ssh-copy-id company_y@192.168.159.136
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
company_y@192.168.159.136's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'company_y@192.168.159.136'"
and check to make sure that only the key(s) you wanted were added.
```

Source: Author

Figure 3.6: Copying the public key to the Web Server

```
nico@nico-virtual-machine:~$ ssh company_y@192.168.159.136
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-91-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Dec 20 02:00:40 PM UTC 2023

  System load:  0.0               Processes:             213
  Usage of /:   30.2% of 9.75GB   Users logged in:       1
  Memory usage: 11%               IPv4 address for ens33: 192.168.159.136
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

44 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Wed Dec 20 10:14:31 2023 from 192.168.159.128
company_y@web-server:~$
```

Source: Author

Figure 3.7: Connecting passwordless via SSH

Finally as the last hardening measure for SSH, we will disable username/password login. Note that we will enable it later for Subchapter 3.5 to demonstrate Fail2Ban. To disable this setting, go back to the `/etc/ssh/sshd_config` file and change `#Password Authentication yes` to `Password Authentication no`. This setting can be found towards the bottom of Figure 3.3. Restart SSH for the change to take effect with `sudo systemctl restart ssh`.

Now, with these steps, SSH is significantly secure, reducing the likelihood of a breach through this access point.

3.2 Using AppArmor

Application Armor is a security module that allows you to set custom restrictions for every application on your system. You can use it to limit everything from network access to read and write permissions. It comes preinstalled on Ubuntu and supports the creation of individual application profiles. [App]

Usually default settings are not necessarily secure, however, in AppArmor's case it comes secure out of the box. It enhances system security by isolating applications confining them to specific resources and actions, thereby reducing the attack surface and minimizing potential damage [App], a crucial aspect of OS Hardening.

We will demonstrate how to check AppArmor's status and creating a custom profile for Apache2 since we are running it on the Web Server.

First to check if AppArmor is running on the system and see the loaded profile set, we want to input `sudo aa-status` on the terminal. The results can be observed in Figure 3.8

```
company_y@web-server:~$ sudo aa-status
apparmor module is loaded.
31 profiles are loaded.
31 profiles are in enforce mode.
   /snap/snapd/19457/usr/lib/snapd/snap-confine
   /snap/snapd/19457/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
   /snap/snapd/20290/usr/lib/snapd/snap-confine
   /snap/snapd/20290/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
   /usr/bin/man
   /usr/lib/NetworkManager/nm-dhcp-client.action
   /usr/lib/NetworkManager/nm-dhcp-helper
   /usr/lib/connman/scripts/dhclient-script
   /usr/lib/snapd/snap-confine
   /usr/lib/snapd/snap-confine//mount-namespace-capture-helper
   /{,usr/}sbin/dhclient
   lsb_release
   man_filter
   man_groff
   nvidia_modprobe
   nvidia_modprobe//kmod
   snap-update-ns.lxd
   snap.lxd.activate
   snap.lxd.benchmark
   snap.lxd.buginfo
   snap.lxd.check-kernel
   snap.lxd.daemon
   snap.lxd.hook.configure
   snap.lxd.hook.install
   snap.lxd.hook.remove
   snap.lxd.lxc
   snap.lxd.lxc-to-lxd
   snap.lxd.lxd
   snap.lxd.migrate
   snap.lxd.user-daemon
   tcpdump
0 profiles are in complain mode.
0 profiles are in kill mode.
0 profiles are in unconfined mode.
0 processes have profiles defined.
0 processes are in enforce mode.
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
0 processes are in mixed mode.
0 processes are in kill mode.
```

Source: Author

Figure 3.8: AppArmor Status and loaded profile

In the case that AppArmor is not enabled on the OS it can be activated with `sudo systemctl start apparmor`.

Now we will demonstrate how to make a new profile for AppArmor. First, we must create the new profile that we will call **usr.sbin.apache2** with `sudo nano /etc/apparmor.d/usr.sbin.apache2`.

For the sake of saving time a profile for Apache2 is provided below:

```
# Last Modified: Sun Dec 20 17:53:23 2023
#include <tunables/global>

/usr/sbin/apache2 {
  #include <abstractions/base>

  capability chown,
  capability dac_override,
  capability dac_read_search,
  capability fowner,
  capability fsetid,
  capability kill,
  capability setgid,
  capability setuid,
  capability setpcap,
  capability net_bind_service,
  capability sys_chroot,
  capability sys_resource,

  # Site-specific additions and overrides. See local/README for details.
  /var/www/html/ r,
  /var/www/html/** r,
  /var/log/apache2/ r,
  /var/log/apache2/*.log w,
  /etc/apache2/ r,
  /etc/apache2/** r,
  /run/apache2/apache2.pid rw,
  /run/apache2/*.pid rw,
  /proc/*/status r,
  /proc/meminfo r,
  /sys/devices/system/cpu/ r,
  /sys/devices/system/cpu/** r,
  /sys/devices/system/cpu/online r,
}
```

Save and exit the file and then reload the AppArmor profiles with `sudo apparmor_parser -r /etc/apparmor.d/usr.sbin.apache2` and restart Apache2 with `sudo systemctl restart apache2`

We can check if the profile is loaded correctly by typing in `sudo aa-status`. The newly created profile should appear.

3.3 iptables

Netfilter serves as the default firewall in every Linux distribution, and iptables is a command-line utility that facilitates the management of netfilter. It allows users to create custom firewall configurations through shell scripts [Tev20] It enables packet filtering, NAT support and stateful inspection.

For persistent rule management, we recommend using **iptables-persistent**, which can be installed with `sudo apt install iptables-persistent`.

To streamline the firewall configuration process, a bash script is provided below with comments explaining what the commands do.

```
#!/bin/sh
# Empty all rules
sudo iptables -F
sudo iptables -t filter -F
sudo iptables -t filter -X
sudo iptables -t nat -F
sudo iptables -t nat -X
# Block everything by default
sudo iptables -t filter -P INPUT DROP
sudo iptables -t filter -P FORWARD DROP
sudo iptables -t filter -P OUTPUT DROP
# Authorize already established connections
sudo iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -t filter -A INPUT -i lo -j ACCEPT
sudo iptables -t filter -A OUTPUT -o lo -j ACCEPT
sudo iptables -t filter -A FORWARD -o lo -j ACCEPT
# ICMP (Ping)
sudo iptables -t filter -A INPUT -p icmp -j ACCEPT
sudo iptables -t filter -A OUTPUT -p icmp -j ACCEPT
sudo iptables -t filter -A FORWARD -p icmp -j ACCEPT
# SSH
sudo iptables -t filter -A INPUT -p tcp --dport 22 -j ACCEPT
sudo iptables -t filter -A OUTPUT -p tcp --dport 22 -j ACCEPT
# HTTP
sudo iptables -t filter -A OUTPUT -p tcp --dport 80 -j ACCEPT
sudo iptables -t filter -A INPUT -p tcp --dport 80 -j ACCEPT
sudo iptables -t filter -A FORWARD -p tcp --dport 80 -j ACCEPT
#HTTPS
sudo iptables -t filter -A OUTPUT -p tcp --dport 443 -j ACCEPT
sudo iptables -t filter -A INPUT -p tcp --dport 443 -j ACCEPT
sudo iptables -t filter -A FORWARD -p tcp --dport 443 -j ACCEPT
```

This script blocks all incoming and outgoing connections by default, except for those explicitly specified. Ensure to update port numbers if modified, as demonstrated in Subchapter 3.1 with SSH.

3.4 Keeping an OS Up-To-Date and Removing Unnecessary Packages

This section highlights the importance of maintaining an up-to-date operating system for security reasons. Regular updates, including security patches and bug fixes, are crucial to safeguard the system against potential vulnerabilities. Keeping an OS up-to-date is a straightforward process. You mostly have to execute `sudo apt update` and `sudo apt upgrade`

We can also use `sudo apt autoremove`, `sudo rpm -qa`, `sudo deborphan`, if installed, and `sudo apt-get remove $packagename` to remove any unnecessary packages.

3.5 Fail2Ban

Now, let's delve into our final tool of the paper: Fail2Ban. Fail2Ban is an open-source log-parsing application designed to bolster security by dynamically blocking IP addresses displaying suspicious behavior. It proves effective in mitigating brute force attacks on services such as SSH, FTP, and web applications like Apache2 Web Server. [fai]

Fail2Ban takes a proactive approach to prevent unauthorized access by automatically identifying and blocking potential threats. It boasts customizable rules that allow for specific configurations tailored to applications and attack patterns. Additionally, it provides centralized management, enabling the monitoring of multiple services through a unified interface.[fai]

We can quite easily demonstrate Fail2Ban's ability to ban suspicous IP addresses by trying to login via SSH by inputting the password wrong several times. This would imitate a brute force attack of the port.

Begin by installing Fail2Ban with `sudo apt install fail2ban`. Check its status with `sudo systemctl status fail2ban` and ensure it starts at boot with `sudo systemctl start fail2ban`.

Next copy the `jail.conf` file to the `jail.local` file with `sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local`. To avoid accidentally banning known IP addresses, add the IP to the `ignoreip` line. The length of how long an IP will be blocked can be set at `bantime`, how many times an IP address can fail to log in before it gets blocked at `maxretry`, and the time period after which the maxretry counter gets reset at `findtime`. Apply the changes by restarting fail2ban with `sudo systemctl restart fail2ban`

```
# "ignoreip" can be a list of IP addresses, CIDR masks or DNS hosts. Fail2ban
# will not ban a host which matches an address in this list. Several addresses
# can be defined using space (and/or comma) separator.
ignoreip = 192.168.159.136

# External command that will take an tagged arguments to ignore, e.g. <ip>,
# and return true if the IP is to be ignored. False otherwise.
#
# ignorecommand = /path/to/command <ip>
ignorecommand =

# "bantime" is the number of seconds that a host is banned.
bantime  = 10m

# A host is banned if it has generated "maxretry" during the last "findtime"
# seconds.
findtime  = 10m

# "maxretry" is the number of failures before a host get banned.
maxretry = 5

# "maxmatches" is the number of matches stored in ticket (resolvable via tag <matches> in actions).
maxmatches = %(maxretry)s
```

Source: Author

Figure 3.9: Fail2Ban configuration

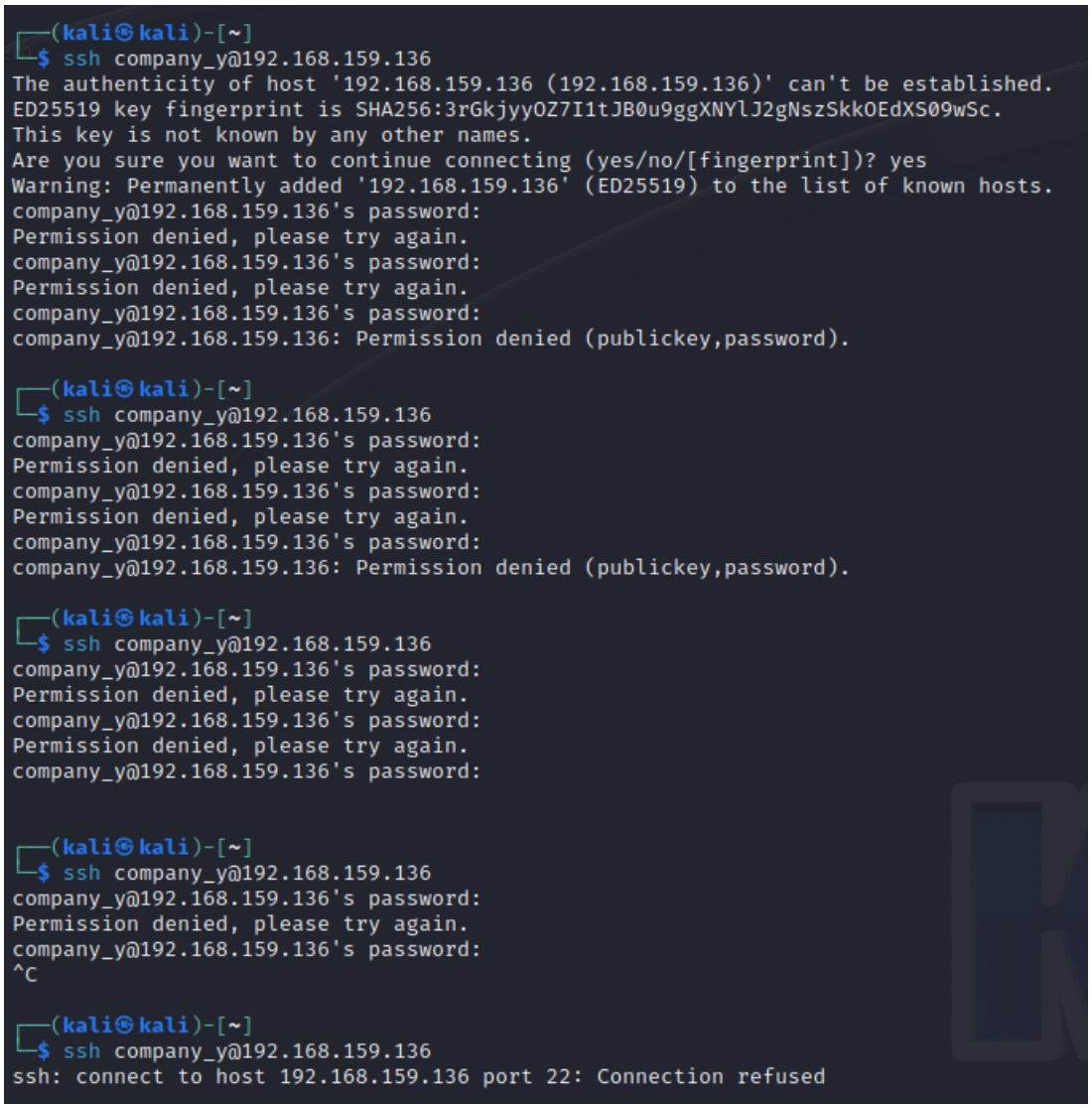
Check the Fail2Ban status with `sudo fail2ban-client status`.There should be one jail for SSH. Examine it in more detail with `sudo fail2ban-client status sshd` as seen in Figure 3.10

```
company_y@web-server:~$ sudo fail2ban-client status
Status
|- Number of jail:      1
`- Jail list:   sshd
company_y@web-server:~$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     0
|  `- File list:        /var/log/auth.log
`- Actions
   |- Currently banned: 0
   |- Total banned:     0
   `- Banned IP list:
```

Source: Author

Figure 3.10: Fail2Ban status and SSH jail status

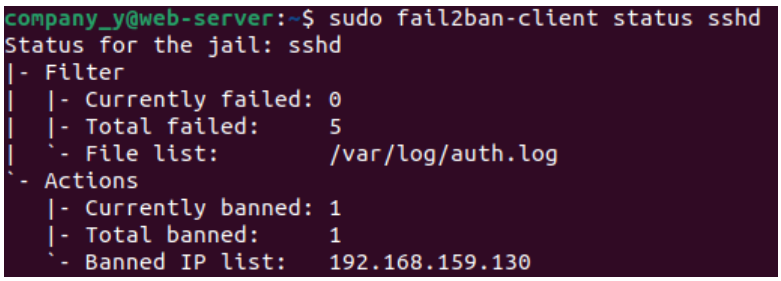
Now we can test fail2ban by attempting multiple incorrect logins with a different virtual machine. Specifically a Kali Linux virtual machine with the IP address 192.168.159.130. After several login attempts, the port will be blocked, as shown in Figure 3.11.



Source: Author

Figure 3.11: Login attempt and block

Now when we check at the Web Server the status of the SSH jail with `sudo fail2ban-client status sshd`. The IP address `192.168.159.130` was banned, as illustrated in Figure 3.12



Source: Author

Figure 3.12: The IP address is banned

Chapter 4

Conclusion

This paper has shed light on the critical importance of system hardening, particularly in the context of the Linux operating system. It has emphasized that the quest for absolute security in computer systems is challenging due to the complex nature of managing multiple programs and applications, some of which originate from third-party sources. System hardening, which involves reducing the attack surface, configuring secure settings, and addressing potential vulnerabilities, is presented as a potent strategy to enhance system security.

The paper has also highlighted the consequences of insufficient system hardening, including data breaches, financial losses, regulatory fines, and damage to reputation. It has stressed the significance of staying informed about evolving threats and vulnerabilities to mitigate risks effectively.

Furthermore, the paper delves into the specific threats that target the Linux operating system, including insecure default settings, malware, weak passwords, privilege escalation, denial of service attacks, and other vulnerabilities. These threats pose significant challenges and necessitate robust security measures.

The section on Linux OS hardening provides practical tools to enhance the security of Linux systems, such as AppArmor, iptables and Fail2Ban amongst others. These measures aim to bolster the security of Linux based systems, making them less vulnerable to a wide range of threats.

In conclusion, this paper emphasizes the critical role of system hardening, particularly in the context of Linux OS, and provides valuable insights into the challenges and solutions for enhancing system security. It serves as a reminder of the ever present need to proactively secure computer systems in our interconnected world.

Bibliography

[Aca23] SSH Academy. Ftp – use of ssh/sftp instead recommended. https://www.ssh.com/academy/ssh/ftp, January 2023. Retrieved 15 December, 2023. 5

[App] Apparmor. https://apparmor.net/. Retrieved 15 December, 2023. 11

[BLLB+23] Michael Bartock, Suzanne Lightman, Ya-Shia Li-Baboud, James McCarthy, Karen Reczek, Joseph Brule, Karri Meldorf, Doug Northrip, Arthur Scholz, and Theresa Suloway. Foundational pnt profile: Applying the cybersecurity framework for the responsible use of positioning, navigation, and timing (pnt) services. Technical Report NIST IR 8323r1, National Institute of Standards and Technology, Gaithersburg, MD, January 2023. 4

[Cen23] Australian Signals Directorate's Australian Cyber Security Centre. Guidelines for system hardening. https://www.cyber.gov.au/resources-business-and-government/essential-cyber-security/ism/cyber-security-guidelines/guidelines-system-hardening, Sep 2023. Retrieved 2 October, 2023. 3

[Cog23] Henry Coggill. What is system hardening? essential checklists from os to applications. https://ubuntu.com/blog/what-is-system-hardening-definition-and-best-practices, Mar 2023. Retrieved 2 October, 2023. 1, 2, 3

[Dan23] Brett Daniel. System hardening: An easy-to-understand overview. https://www.trentonsystems.com/blog/system-hardening-overview, Feb 2023. Retrieved 2 October, 2023. 2

[fai] fail2ban. Fail2ban. https://github.com/fail2ban/fail2ban. 15

[fIS] Center for Internet Security. Cis benchmarks. https://www.cisecurity.org/cis-benchmarks. Retrieved 2 October, 2023. 2

[Fou] The Apache Software Foundation. Impact levels for apache httpd. https://httpd.apache.org/security/impact_levels.html. Retrieved 15 December, 2023. 5

[Hou23] The White House. Budget of the u.s government, information technology and cybersecurity funding. https://www.whitehouse.gov/wp-content/

uploads/2023/03/ap_14_it_fy2024.pdf, March 2023. Retrieved 2 October, 2023. 2

[Jog17] Martin Jogi. Establishing, implementing and auditing linux operating system hardening standard for security compliance. *University of Tartu, Tartu*, 2017. 2

[MTFM17] Robert Mendick, James Titcomb, Ben Farmer, and Cara McGoogan. Russian-linked cyber gang blamed for nhs computer hack using bug stolen from us spy agency. https://www.telegraph.co.uk/news/2017/05/12/russian-linked-cyber-gang-shadow-brokers-blamed-nhs-computer/, May 2017. Retrieved 4 October, 2023. 4

[New23] The Hacker News. Looney tunables: New linux flaw enables privilege escalation on major distributions. https://thehackernews.com/2023/10/looney-tunables-new-linux-flaw-enables.html, Oct 2023. Retrieved 4 October, 2023. 4

[NMZL14] Shuangxia Niu, Jiansong Mo, Zhigang Zhang, and Zhuo Lv. Overview of linux vulnerabilities. In *2nd International Conference on Soft Computing in Information Communication Technology*, pages 225–228. Atlantis Press, 2014. 2, 4

[oST] National Institute of Standards and Technology. Nvd - cve-2014-6271 detail. https://nvd.nist.gov/vuln/detail/CVE-2014-6271. Retrieved 15 December, 2023. 5

[oST17] National Institute of Standards and Technology. National checklist program: Csrc. https://csrc.nist.gov/Projects/National-Checklist-Program, Feb 2017. Retrieved 2 October, 2023. 2

[OWA] OWASP. System hardening. https://owasp.org/www-project-developer-guide/draft/09-secure-environment/02-system-hardening. Retrieved 3 October, 2023. 2

[PP10] Prof Patra and P.L. Pradhan. Hardening of unix operating system. *International Journal of Computer and Communication Technology*, pages 71–84, 01 2010.

[SJT08] Karen Scarfone, Wayne A. Jansen, and Miles Tracy. Guide to general server security. Technical Report NIST SP 800-123, National Institute of Standards and Technology, Gaithersburg, MD, July 2008. 5

[SPT+23] Keith Stouffer, Michael Pease, CheeYee Tang, Timothy Zimmerman, Victoria Pillitteri, Suzanne Lightman, Adam Hahn, Stephanie Saravia, Aslam Sherule, and Michael Thompson. Guide to operational technology (ot) security. Technical Report NIST SP 800-82 Rev. 3, National Institute of Standards and Technology, Gaithersburg, MD, September 2023. 5

[SS23] Raphael Satter and Zeba Siddiqui. Chinese hackers stole emails from us state dept in microsoft breach, senate staffer says. `http://tinyurl.com/reuters-microsoft-breach`, Sep 2023. Retrieved 2 October, 2023. 3

[Tev20] Donald A Tevault. *Mastering Linux Security and Hardening: Protect your Linux systems from intruders, malware attacks, and other cyber threats*. Packt Publishing Ltd, 2020. 3, 4, 5, 8, 9, 14