How To Secure A Linux Server

An evolving how-to guide for securing a Linux server that, hopefully, also teaches you a little about security and why it matters.

Table of Contents

(TOC made with nGitHubTOC)

Introduction

Guide Objective

This guides purpose is to teach you how to secure a Linux server.

There are a lot of things you can do to secure a Linux server and this guide will attempt to cover as many of them as possible. More topics/material will be added as I learn, or as folks contribute.

Ansible playbooks of this guide are available at How To Secure A Linux Server With Ansible by moltenbit.

(Table of Contents)

Why Secure Your Server

I assume you're using this guide because you, hopefully, already understand why good security is important. That is a heavy topic onto itself and breaking it down is out-of-scope for this guide. If you don't know the answer to that question, I advise you research it first.

At a high level, the second a device, like a server, is in the public domain -- i.e. visible to the outside world -- it becomes a target for bad-actors. An unsecured device is a playground for bad-actors who want access to your data, or to use your server as another node for their large-scale DDOS attacks.

What's worse is, without good security, you may never know if your server has been compromised. A bad-actor may have gained unauthorized access to your server and copied your data without changing anything, so you'd never know. Or your server may have been part of a DDOS attack, and you wouldn't know. Look at many of the large scale data breaches in the news -- the companies often did not discover the data leak or intrusion until long after the bad-actors were gone.

Contrary to popular belief, bad-actors don't always want to change something or lock you out of your data for money. Sometimes they just want the data on your server for their data warehouses (there is big money in big data) or to covertly use your server for their nefarious purposes.

(Table of Contents)

Why Yet Another Guide

This guide may appear duplicative/unnecessary because there are countless articles online that tell you how to secure Linux, but the information is spread across different articles, that cover different things, and in different ways. Who has time to scour through hundreds of articles?

As I was going through research for my Debian build, I kept notes. At the end I realized that, along with what I already knew, and what I was learning, I had the makings of a how-to guide. I figured I'd put it online to hopefully help others **learn**, and **save time**.

I've never found one guide that covers everything -- this guide is my attempt.

Many of the things covered in this guide may be rather basic/trivial, but most of us do not install Linux every day, and it is easy to forget those basic things.

(Table of Contents)

Other Guides

There are many guides provided by experts, industry leaders, and the distributions themselves. It is not practical, and sometimes against copyright, to include everything from those guides. I recommend you check them out before starting with this guide.

- The Center for Internet Security (CIS) provides benchmarks that are exhaustive, industry trusted, step-by-step instructions for securing many flavors of Linux. Check their About Us page for details. My recommendation is to go through this guide (the one you're reading here) first and THEN CIS's guide. That way their recommendations will trump anything in this guide.
- For distribution specific hardening/security guides, check your distributions documentation.
- https://security.utexas.edu/os-hardening-checklist/linux-7 - Red Hat Enterprise Linux 7 Hardening Checklist
- https://cloudpro.zone/index.php/2018/01/18/debian-9-3-server-setup-guide-part-1/ - # Debian 9.3 server setup guide
- https://blog.vigilcode.com/2011/04/ubuntu-server-initial-security-quick-secure-setup-part-i/ - Ubuntu Server Initial Security guide
- https://www.tldp.org/LDP/sag/html/index.html
- https://seifried.org/lasg/
- https://news.ycombinator.com/item?id=19178964
- https://wiki.archlinux.org/index.php/Security - many folks have also recommended this one
- https://securecompliance.co/linux-server-hardening-checklist/

(Table of Contents)

To Do / To Add

- [ ] Custom Jails for Fail2ban
- [ ] MAC (Mandatory Access Control) and Linux Security Modules (LSMs)
  - https://wiki.archlinux.org/index.php/security#Mandatory_access_control
  - Security-Enhanced Linux / SELinux
    - https://en.wikipedia.org/wiki/Security-Enhanced_Linux
    - https://linuxtechlab.com/beginners-guide-to-selinux/
    - https://linuxtechlab.com/replicate-selinux-policies-among-linux-machines/
    - https://teamignition.us/how-to-stop-being-a-scrub-and-learn-to-use-selinux.html
  - AppArmor
    - https://wiki.archlinux.org/index.php/AppArmor
    - https://security.stackexchange.com/questions/29378/comparison-between-apparmor-and-selinux
    - http://www.insanitybit.com/2012/06/01/why-i-like-apparmor-more-than-selinux-5/
- [ ] disk encryption
- [ ] Rkhunter and chrootkit
  - http://www.chkrootkit.org/
  - http://rkhunter.sourceforge.net/

    - https://www.cyberciti.biz/faq/howto-check-linux-rootkist-with-detectors-software/
    - https://www.tecmint.com/install-rootkit-hunter-scan-for-rootkits-backdoors-in-linux/
- [ ] shipping/backing up logs - https://news.ycombinator.com/item?id=19178681
- [ ] CIS-CAT - https://learn.cisecurity.org/cis-cat-landing-page
- [ ] debsums - https://blog.sleeplessbeastie.eu/2015/03/02/how-to-verify-installed-packages/

(Table of Contents)

Guide Overview

About This Guide

This guide...

- ...**is** a work in progress.
- ...**is** focused on **at-home** Linux servers. All of the concepts/recommendations here apply to larger/professional environments but those use-cases call for more advanced and specialized configurations that are out-of-scope for this guide.
- ...**does not** teach you about Linux, how to install Linux, or how to use it. Check https://linuxjourney.com/ if you're new to Linux.
- ...**is** meant to be Linux distribution agnostic.
- ...**does not** teach you everything you need to know about security nor does it get into all aspects of system/server security. For example, physical security is out of scope for this guide.
- ...**does not** talk about how programs/tools work, nor does it delve into their nook and crannies. Most of the programs/tools this guide references are very powerful and highly configurable. The goal is to cover the bare necessities -- enough to whet your appetite and make you hungry enough to want to go and learn more.
- ...**aims** to make it easy by providing code you can copy-and-paste. You might need to modify the commands before you paste so keep your favorite text editor handy.
- ...**is** organized in an order that makes logical sense to me -- i.e. securing SSH before installing a firewall. As such, this guide is intended to be followed in the order it is presented, but it is not necessary to do so. Just be careful if you do things in a different order -- some sections require previous sections to be completed.

(Table of Contents)

My Use-Case

There are many types of servers and different use-cases. While I want this guide to be as generic as possible, there will be some things that may not apply to all/other use-cases. Use your best judgement when going through this guide.

To help put context to many of the topics covered in this guide, my use-case/configuration is:

- A desktop class computer...
- With a single NIC...
- Connected to a consumer grade router...
- Getting a dynamic WAN IP provided by the ISP...
- With WAN+LAN on IPV4...
- And LAN using NAT...
- That I want to be able to SSH to remotely from unknown computers and unknown locations (i.e. a friend's house).

(Table of Contents)

Editing Configuration Files - For The Lazy

I am very lazy and do not like to edit files by hand if I don't need to. I also assume everyone else is just like me. :)

So, when and where possible, I have provided `code` snippets to quickly do what is needed, like add or change a line in a configuration file.

The `code` snippets use basic commands like `echo`, `cat`, `sed`, `awk`, and `grep`. How the `code` snippets work, like what each command/part does, is out of scope for this guide -- the `man` pages are your friend.

Note: The `code` snippets do not validate/verify the change went through -- i.e. the line was actually added or changed. I'll leave the verifying part in your capable hands. The steps in this guide do include taking backups of all files that will be changed.

Not all changes can be automated with `code` snippets. Those changes need good, old-fashioned, manual editing. For example, you can't just append a line to an INI type file. Use your favorite Linux text editor.

(Table of Contents)

Contributing

I wanted to put this guide on GitHub to make it easy to collaborate. The more folks that contribute, the better and more complete this guide will become.

To contribute you can fork and submit a pull request or submit a new issue.

(Table of Contents)

Before You Start

Identify Your Principles

Before you start you will want to identify what your Principles are. What is your threat model? Some things to think about:

- Why do you want to secure your server?
- How much security do you want or not want?
- How much convenience are you willing to compromise for security and vice-versa?
- What are the threats you want to protect against? What are the specifics to your situation? For example:
  - Is physical access to your server/network a possible attack vector?
  - Will you be opening ports on your router so you can access your server from outside your home?
  - Will you be hosting a file share on your server that will be mounted on a desktop class machine? What is the possibility of the desktop machine getting infected and, in turn, infecting the server?
- Do you have a means of recovering if your security implementation locks you out of your own server? For example, you disabled root login or password protected GRUB.

These are just **a few things** to think about. Before you start securing your server you will want to understand what you're trying to protect against and why so you know what you need to do.

(Table of Contents)

Picking A Linux Distribution

This guide is intended to be distribution agnostic so users can use any distribution they want. With that said, there are a few things to keep in mind:

You want a distribution that...

- ...**is stable**. Unless you like debugging issues at 2 AM, you don't want an unattended upgrade, or a manual package/system update, to render your server inoperable. But this also means you're okay with not running the latest, greatest, bleeding edge software.
- ...**stays up-to-date with security patches**. You can secure everything on your server, but if the core OS or applications you're running have known vulnerabilities, you'll never be safe.
- ...**you're familiar with.** If you don't know Linux, I would advise you play around with one before you try to secure it. You should be comfortable with it and know your way around, like how to install software, where configuration files are, etc...
- ...**is well-supported.** Even the most seasoned admin needs help every now and then. Having a place to go for help will save your sanity.

(Table of Contents)

Installing Linux

Installing Linux is out-of-scope for this guide because each distribution does it differently and the installation instructions are usually well documented. If you need help, start with your distribution's documentation. Regardless of the distribution, the high-level process usually goes like so:

1. download the ISO
2. burn/copy/transfer it to your install medium (e.g. a CD or USB stick)
3. boot your server from your install medium
4. follow the prompts to install

Where applicable, use the expert install option so you have tighter control of what is running on your server. **Only install what you absolutely need.** I, personally, do not install anything other than SSH. Also, tick the Disk Encryption option.

(Table of Contents)

Pre/Post Installation Requirements

- If you're opening ports on your router so you can access your server from the outside, disable the port forwarding until your system is up and secured.
- Unless you're doing everything physically connected to your server, you'll need remote access so be sure SSH works.
- Keep your system up-to-date (i.e. `sudo apt update && sudo apt upgrade` on Debian based systems).
- Make sure you perform any tasks specific to your setup like:
  - Configuring network
  - Configuring mount points in `/etc/fstab`
  - Creating the initial user accounts
  - Installing core software you'll want like `man`
  - Etc...
- Your server will need to be able to send e-mails so you can get important security alerts. If you're not setting up a mail server check Gmail and Exim4 As MTA With Implicit TLS.
- I would also recommend you **read** through the CIS Benchmarks before you start with this guide just to digest/understand what they have to say. My recommendation is to go through this guide (the one you're reading here) first and THEN CIS's guide. That way their recommendations will trump anything in this guide.

(Table of Contents)

Other Important Notes

- This guide is being written and tested on Debian. Most things below should work on other distributions. If you find something that does not, please contact me. The main thing that separates each distribution will be its package management system. Since I use Debian, I will provide the appropriate `apt` commands that should work on all Debian based distributions. If someone is willing to provide the respective commands for other distributions, I will add them.
- File paths and settings also may differ slightly -- check with your distribution's documentation if you have issues.
- Read the whole guide before you start. Your use-case and/or principals may call for not doing something or for changing the order.
- Do not **blindly** copy-and-paste without understanding what you're pasting. Some commands will need to be modified for your needs before they'll work -- usernames for example.

(Table of Contents)

Using Ansible playbooks to secure your Linux Server

Ansible playbooks of this guide are available at How To Secure A Linux Server With Ansible.

Make sure to edit the variables according to your needs and read all tasks beforehand to confirm it does not break your system. After running the playbooks ensure that all settings are configured to your needs!

1. Install Ansible
2. git clone How To Secure A Linux Server With Ansible
3. Create SSH-Public/Private-Keys

```
ssh-keygen -t ed25519
```

5. Change all variables in *group_vars/variables.yml* according to your needs.
6. Enable SSH root access before running the playbooks:

```
nano /etc/ssh/sshd_config
[...]
PermitRootLogin yes
[...]
```

7. Recommended: configure static IP address on your system.
8. Add your systems IP address to *hosts.yml*.

Run the requirements playbook using the root password you specified while installing the server:

```
ansible-playbook --inventory hosts.yml --ask-pass requirements-playbook.yml
```

Run the main playbook with the new users password you specified in the *variables.yml* file:

```
ansible-playbook --inventory hosts.yml --ask-pass main-playbook.yml
```

If you need to run the playbooks multiple times remember to use the SSH key and the new SSH port:

```
ansible-playbook --inventory hosts.yml -e ansible_ssh_port=SSH_PORT --key-file /PATH/TO/SSH/KEY main-playbook.yml
```

(Table of Contents)

The SSH Server

Important Note Before You Make SSH Changes

It is highly advised you keep a 2nd terminal open to your server **before you make and apply SSH configuration changes**. This way if you lock yourself out of your 1st terminal session, you still have one session connected so you can fix it.

Thank you to Sonnenbrand for this idea.

SSH Public/Private Keys

Why

Using SSH public/private keys is more secure than using a password. It also makes it easier and faster, to connect to our server because you don't have to enter a password.

How It Works

Check the references below for more details but, at a high level, public/private keys work by using a pair of keys to verify identity.

1. One key, the **public** key, **can only encrypt data**, not decrypt it
2. The other key, the **private** key, can decrypt the data

For SSH, a public and private key is created on the client. You want to keep both keys secure, especially the private key. Even though the public key is meant to be public, it is wise to make sure neither keys fall in the wrong hands.

When you connect to an SSH server, SSH will look for a public key that matches the client you're connecting from in the file `~/.ssh/authorized_keys` on the server you're connecting to. Notice the file is in the **home folder** of the ID you're trying to connect to. So, after creating the public key, you need to append it to `~/.ssh/authorized_keys`. One approach is to copy it to a USB stick and physically transfer it to the server. Another approach is to use `ssh-copy-id` to transfer and append the public key.

After the keys have been created and the public key has been appended to `~/.ssh/authorized_keys` on the host, SSH uses the public and private keys to verify identity and then establish a secure connection. How identity is verified is a complicated process but Digital Ocean has a very nice write-up of how it works. At a high level, identity is verified by the server encrypting a challenge message with the public key, then sending it to the client. If the client cannot decrypt the challenge message with the private key, the identity can't be verified and a connection will not be established.

They are considered more secure because you need the private key to establish an SSH connection. If you set `PasswordAuthentication no` in `/etc/ssh/sshd_config`, then SSH won't let you connect without the private key.

You can also set a pass-phrase for the keys which would require you to enter the key pass-phrase when connecting using public/private keys. Keep in mind doing this means you can't use the key for automation because you'll have no way to send the passphrase in your scripts. `ssh-agent` is a program that is shipped in many Linux distros (and usually already running) that will allow you to hold your unencrypted private key in memory for a configurable duration. Simply run `ssh-add` and it will prompt you for your passphrase. You will not be prompted for your passphrase again until the configurable duration has passed.

We will be using Ed25519 keys which, according to https://linux-audit.com/:

> It is using an elliptic curve signature scheme, which offers better security than ECDSA and DSA. At the same time, it also has good performance.

Goals

- Ed25519 public/private SSH keys:
  - private key on your client
  - public key on your server

Notes

- You'll need to do this step for every computer and account you'll be connecting to your server from/as.

References

- https://www.ssh.com/ssh/public-key-authentication
- https://help.ubuntu.com/community/SSH/OpenSSH/Keys
- https://linux-audit.com/using-ed25519-openssh-keys-instead-of-dsa-rsa-ecdsa/
- https://www.digitalocean.com/community/tutorials/understanding-the-ssh-encryption-and-connection-process
- https://wiki.archlinux.org/index.php/SSH_Keys
- https://www.ssh.com/ssh/copy-id
- `man ssh-keygen`
- `man ssh-copy-id`
- `man ssh-add`

Steps

1. From the computer you're going to use to connect to your server, **the client**, not the server itself, create an Ed25519 key with `ssh-keygen`:

```
ssh-keygen -t ed25519
```

```
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/user/.ssh/id_ed25519):
Created directory '/home/user/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_ed25519.
Your public key has been saved in /home/user/.ssh/id_ed25519.pub.
The key fingerprint is:
SHA256:F44D4dr2zoHqgj0i2iVIHQ32uk/Lx4P+raayEAQjlcs user@client
The key's randomart image is:
+--[ED25519 256]--+
|xxxx  x          |
|o.o +. .         |
| o o oo   .      |
|. E oo . o .     |
| o o. o S o      |
|... .. o o       |
|.+....+ o        |
|+.=++o.B..       |
|+..=**=o=.       |
+----[SHA256]-----+
```

   Note: If you set a passphrase, you'll need to enter it every time you connect to your server using this key, unless you're using `ssh-agent`.

2. Now you need to **append** the public key `~/.ssh/id_ed25519.pub` from your client to the `~/.ssh/authorized_keys` file on your server. Since we're presumable still at home on the LAN, we're probably safe from MIM attacks, so we will use `ssh-copy-id` to transfer and append the public key:

```
ssh-copy-id user@server
```

```
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/user/.ssh/id_ed25519.pub"
The authenticity of host 'host (192.168.1.96)' can't be established.
ECDSA key fingerprint is SHA256:QaDQb/X0XyVlogh87sDXE7MR8YIK7ko4wS5hXjRySJE.
Are you sure you want to continue connecting (yes/no)? yes
/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
user@host's password:

Number of key(s) added: 1

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

Now would be a good time to perform any tasks specific to your setup.

(Table of Contents)

Create SSH Group For AllowGroups

Why

To make it easy to control who can SSH to the server. By using a group, we can quickly add/remove accounts to the group to quickly allow or not allow SSH access to the server.

How It Works

We will use the AllowGroups option in SSH's configuration file `/etc/ssh/sshd_config` to tell the SSH server to only allow users to SSH in if they are a member of a certain UNIX group. Anyone not in the group will not be able to SSH in.

Goals

- a UNIX group that we'll use in Secure `/etc/ssh/sshd_config` to limit who can SSH to the server

Notes

- This is a prerequisite step to support the `AllowGroup` setting set in Secure `/etc/ssh/sshd_config` .

References

- `man groupadd`
- `man usermod`

Steps

1. Create a group:

```
sudo groupadd sshusers
```

2. Add account(s) to the group:

```
sudo usermod -a -G sshusers user1
sudo usermod -a -G sshusers user2
sudo usermod -a -G sshusers ...
```

You'll need to do this for every account on your server that needs SSH access.

(Table of Contents)

Secure `/etc/ssh/sshd_config`

Why

SSH is a door into your server. This is especially true if you are opening ports on your router so you can SSH to your server from outside your home network. If it is not secured properly, a bad-actor could use it to gain unauthorized access to your system.

How It Works

`/etc/ssh/sshd_config` is the default configuration file that the SSH server uses. We will use this file to tell what options the SSH server should use.

Goals

- a secure SSH configuration

Notes

- Make sure you've completed Create SSH Group For AllowGroups first.

References

- Mozilla's OpenSSH guidelines for OpenSSH 6.7+ at https://infosec.mozilla.org/guidelines/openssh#modern-openssh-67
- https://linux-audit.com/audit-and-harden-your-ssh-configuration/
- https://www.ssh.com/ssh/sshd_config/
- https://www.techbrown.com/harden-ssh-secure-linux-vps-server/ (broken; try http://web.archive.org/web/20200413100933/https://www.techbrown.com/harden-ssh-secure-linux-vps-server/)
- https://serverfault.com/questions/660160/openssh-difference-between-internal-sftp-and-sftp-server/660325
- `man sshd_config`
- Thanks to than0s for how to find duplicate settings.

Steps

1. Make a backup of OpenSSH server's configuration file `/etc/ssh/sshd_config` and remove comments to make it easier to read:

```
sudo cp --archive /etc/ssh/sshd_config /etc/ssh/sshd_config-COPY-$(date +"%Y%m%d%H%M%S")
sudo sed -i -r -e '/^#|^$/ d' /etc/ssh/sshd_config
```

2. Edit `/etc/ssh/sshd_config` then find and edit or add these settings that should be applied regardless of your configuration/setup:

   Note: SSH does not like duplicate contradicting settings. For example, if you have `ChallengeResponseAuthentication no` and then `ChallengeResponseAuthentication yes`, SSH will respect the first one and ignore the second. Your `/etc/ssh/sshd_config` file may already have some of the settings/lines below. To avoid issues you will need to manually go through your `/etc/ssh/sshd_config` file and address any duplicate contradicting settings.

```
##################################################################################################
# start settings from https://infosec.mozilla.org/guidelines/openssh#modern-openssh-67 as of 2019-01-01
##################################################################################################

# Supported HostKey algorithms by order of preference.
HostKey /etc/ssh/ssh_host_ed25519_key
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key

KexAlgorithms curve25519-sha256@libssh.org,ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256,diffie-hellman-
group-exchange-sha256

Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-gcm@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr

MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha2-512,hmac-sha2-256,umac-128@openssh.com

# LogLevel VERBOSE logs user's key fingerprint on login. Needed to have a clear audit track of which key was using to
log in.
LogLevel VERBOSE

# Use kernel sandbox mechanisms where possible in unprivileged processes
# Systrace on OpenBSD, Seccomp on Linux, seatbelt on MacOSX/Darwin, rlimit elsewhere.
# Note: This setting is deprecated in OpenSSH 7.5 (https://www.openssh.com/txt/release-7.5)
# UsePrivilegeSeparation sandbox

##################################################################################################
# end settings from https://infosec.mozilla.org/guidelines/openssh#modern-openssh-67 as of 2019-01-01
##################################################################################################

# don't let users set environment variables
PermitUserEnvironment no

# Log sftp level file access (read/write/etc.) that would not be easily logged otherwise.
Subsystem sftp  internal-sftp -f AUTHPRIV -l INFO

# only use the newer, more secure protocol
Protocol 2

# disable X11 forwarding as X11 is very insecure
# you really shouldn't be running X on a server anyway
X11Forwarding no

# disable port forwarding
AllowTcpForwarding no
AllowStreamLocalForwarding no
GatewayPorts no
PermitTunnel no

# don't allow login if the account has an empty password
PermitEmptyPasswords no

# ignore .rhosts and .shosts
IgnoreRhosts yes

# verify hostname matches IP
UseDNS yes

Compression no
TCPKeepAlive no
AllowAgentForwarding no
PermitRootLogin no
```

```
# don't allow .rhosts or /etc/hosts.equiv
HostbasedAuthentication no

# https://github.com/imthenachoman/How-To-Secure-A-Linux-Server/issues/115
HashKnownHosts yes
```

3. Then **find and edit or add** these settings, and set values as per your requirements:

Setting	Valid Values	Example	Description	Notes
AllowGroups	local UNIX group name	`AllowGroups sshusers`	group to allow SSH access to	
ClientAliveCountMax	number	`ClientAliveCountMax 0`	maximum number of client alive messages sent without response	
ClientAliveInterval	number of seconds	`ClientAliveInterval 300`	timeout in seconds before a response request	
ListenAddress	space separated list of local addresses	○ `ListenAddress 0.0.0.0` ○ `ListenAddress 192.168.1.100`	local addresses `sshd` should listen on	See Issue #1 for important details.
LoginGraceTime	number of seconds	`LoginGraceTime 30`	time in seconds before login times-out	
MaxAuthTries	number	`MaxAuthTries 2`	maximum allowed attempts to login	
MaxSessions	number	`MaxSessions 2`	maximum number of open sessions	
MaxStartups	number	`MaxStartups 2`	maximum number of login sessions	
PasswordAuthentication	`yes` or `no`	`PasswordAuthentication no`	if login with a password is allowed	
Port	any open/available port number	`Port 22`	port that `sshd` should listen on	

Check `man sshd_config` for more details what these settings mean.

4. Make sure there are no duplicate settings that contradict each other. The below command should not have any output.

```
awk 'NF && $1!~/^(#|HostKey)/{print $1}' /etc/ssh/sshd_config | sort | uniq -c | grep -v ' 1 '
```

5. Restart ssh:

```
sudo service sshd restart
```

6. You can check verify the configurations worked with `sshd -T` and verify the output:

```
sudo sshd -T
```

```
port 22
addressfamily any
listenaddress [::]:22
listenaddress 0.0.0.0:22
usepam yes
logingracetime 30
x11displayoffset 10
maxauthtries 2
maxsessions 2
clientaliveinterval 300
clientalivecountmax 0
streamlocalbindmask 0177
permitrootlogin no
ignorerhosts yes
```

```
ignoreuserknownhosts no
hostbasedauthentication no
...
subsystem sftp internal-sftp -f AUTHPRIV -l INFO
maxstartups 2:30:2
permittunnel no
ipqos lowdelay throughput
rekeylimit 0 0
permitopen any
```

(Table of Contents)

Remove Short Diffie-Hellman Keys

Why

Per Mozilla's OpenSSH guidelines for OpenSSH 6.7+, "all Diffie-Hellman moduli in use should be at least 3072-bit-long".

The Diffie-Hellman algorithm is used by SSH to establish a secure connection. The larger the moduli (key size) the stronger the encryption.

Goals

- remove all Diffie-Hellman keys that are less than 3072 bits long

References

- Mozilla's OpenSSH guidelines for OpenSSH 6.7+ at https://infosec.mozilla.org/guidelines/openssh#modern-openssh-67
- https://infosec.mozilla.org/guidelines/key_management
- `man moduli`

Steps

1. Make a backup of SSH's moduli file `/etc/ssh/moduli`:

```
sudo cp --archive /etc/ssh/moduli /etc/ssh/moduli-COPY-$(date +"%Y%m%d%H%M%S")
```

2. Remove short moduli:

```
sudo awk '$5 >= 3071' /etc/ssh/moduli | sudo tee /etc/ssh/moduli.tmp
sudo mv /etc/ssh/moduli.tmp /etc/ssh/moduli
```

(Table of Contents)

2FA/MFA for SSH

Why

Even though SSH is a pretty good security guard for your doors and windows, it is still a visible door that bad-actors can see and try to brute-force in. Fail2ban will monitor for these brute-force attempts but there is no such thing as being too secure. Requiring two factors adds an extra layer of security.

Using Two-Factor Authentication (2FA) / Multi-Factor Authentication (MFA) requires anyone entering to have **two** keys to enter which makes it harder for bad actors. The two keys are:

1. Their password
2. A 6 digit token that changes every 30 seconds

Without both keys, they won't be able to get in.

Why Not

Many folks might find the experience cumbersome or annoying. And, access to your system is dependent on the accompanying authenticator app that generates the code.

How It Works

On Linux, PAM is responsible for authentication. There are four tasks to PAM that you can read about at https://en.wikipedia.org/wiki/Linux_PAM. This section talks about the authentication task.

When you log into a server, be it directly from the console or via SSH, the door you came through will send the request to the authentication task of PAM and PAM will ask for and verify your password. You can customize the rules each doors use. For example, you could have one set of rules when logging in directly from the console and another set of rules for when logging in via SSH.

This section will alter the authentication rules for when logging in via SSH to require both a password and a 6 digit code.

We will use Google's libpam-google-authenticator PAM module to create and verify a TOTP key. https://fastmail.blog/2016/07/22/how-totp-authenticator-apps-work/ and https://jemurai.com/2018/10/11/how-it-works-totp-based-mfa/ have very good writeups of how TOTP works.

What we will do is tell the server's SSH PAM configuration to ask the user for their password and then their numeric token. PAM will then verify the user's password and, if it is correct, then it will route the authentication request to libpam-google-authenticator which will ask for and verify your 6 digit token. If, and only if, everything is good will the authentication succeed and user be allowed to log in.

Goals

- 2FA/MFA enabled for all SSH connections

Notes

- Before you do this, you should have an idea of how 2FA/MFA works and you'll need an authenticator app on your phone to continue.
- We'll use google-authenticator-libpam.
- With the below configuration, a user will only need to enter their 2FA/MFA code if they are logging on with their password but **not** if they are using SSH public/private keys. Check the documentation on how to change this behavior to suite your requirements.

References

- https://github.com/google/google-authenticator-libpam
- https://en.wikipedia.org/wiki/Linux_PAM
- https://en.wikipedia.org/wiki/Time-based_One-time_Password_algorithm
- https://fastmail.blog/2016/07/22/how-totp-authenticator-apps-work/
- https://jemurai.com/2018/10/11/how-it-works-totp-based-mfa/

Steps

1. Install it libpam-google-authenticator.

   On Debian based systems:

   ```
   sudo apt install libpam-google-authenticator
   ```

2. **Make sure you're logged in as the ID you want to enable 2FA/MFA for** and **execute** `google-authenticator` to create the necessary token data:

   ```
   google-authenticator
   ```

   ```
   Do you want authentication tokens to be time-based (y/n) y
   https://www.google.com/chart?
   chs=200x200&chld=M|0&cht=qr&chl=otpauth://totp/user@host%3Fsecret%3DR4ZWX34FQKZROVX7AGLJ64684Y%26issuer%3Dhost

   ...

   Your new secret key is: R3NVX3FFQKZROVX7AGLJUGGESY
   Your verification code is 751419
   Your emergency scratch codes are:
     12345678
     90123456
     78901234
     56789012
     34567890

   Do you want me to update your "/home/user/.google_authenticator" file (y/n) y

   Do you want to disallow multiple uses of the same authentication
   token? This restricts you to one login about every 30s, but it increases
   your chances to notice or even prevent man-in-the-middle attacks (y/n) Do you want to disallow multiple uses of the
   same authentication
   token? This restricts you to one login about every 30s, but it increases
   your chances to notice or even prevent man-in-the-middle attacks (y/n) y

   By default, tokens are good for 30 seconds. In order to compensate for
   ```

```
possible time-skew between the client and the server, we allow an extra
token before and after the current time. If you experience problems with
poor time synchronization, you can increase the window from its default
size of +-1min (window size of 3) to about +-4min (window size of
17 acceptable tokens).
Do you want to do so? (y/n) y

If the computer that you are logging into isn't hardened against brute-force
login attempts, you can enable rate-limiting for the authentication module.
By default, this limits attackers to no more than 3 login attempts every 30s.
Do you want to enable rate-limiting (y/n) y
```

Notice this is **not run as root**.

Select default option (y in most cases) for all the questions it asks and remember to save the emergency scratch codes.

3. Make a backup of PAM's SSH configuration file `/etc/pam.d/sshd` :

```
sudo cp --archive /etc/pam.d/sshd /etc/pam.d/sshd-COPY-$(date +"%Y%m%d%H%M%S")
```

4. Now we need to enable it as an authentication method for SSH by adding this line to `/etc/pam.d/sshd` :

```
auth       required     pam_google_authenticator.so nullok
```

Note: Check here for what `nullok` means.

For the lazy:

```
echo -e "\nauth       required     pam_google_authenticator.so nullok         # added by $(whoami) on $(date +"%Y-%m-%d @ %
```

5. Tell SSH to leverage it by adding or editing this line in `/etc/ssh/sshd_config` :

```
ChallengeResponseAuthentication yes
```

For the lazy:

```
sudo sed -i -r -e "s/^(challengeresponseauthentication .*)$/# \1         # commented by $(whoami) on $(date +"%Y-%m-%d @ %H
echo -e "\nChallengeResponseAuthentication yes         # added by $(whoami) on $(date +"%Y-%m-%d @ %H:%M:%S")" | sudo tee -
```

6. Restart ssh:

```
sudo service sshd restart
```

(Table of Contents)

The Basics

Limit Who Can Use sudo

Why

sudo lets accounts run commands as other accounts, including **root**. We want to make sure that only the accounts we want can use sudo.

Goals

- sudo privileges limited to those who are in a group we specify

Notes

- Your installation may have already done this, or may already have a special group intended for this purpose so check first.
  - Debian creates the sudo group. To view users that are part of this group (thus have sudo privileges):

```
cat /etc/group | grep "sudo"
```

- RedHat creates the wheel group
- See #39 for a note on some distributions making it so `sudo` does not require a password. Thanks to sbrl for sharing.

Steps

1. Create a group:

```
sudo groupadd sudousers
```

2. Add account(s) to the group:

```
sudo usermod -a -G sudousers user1
sudo usermod -a -G sudousers user2
sudo usermod -a -G sudousers  ...
```

   You'll need to do this for every account on your server that needs sudo privileges.

3. Make a backup of the sudo's configuration file `/etc/sudoers`:

```
sudo cp --archive /etc/sudoers /etc/sudoers-COPY-$(date +"%Y%m%d%H%M%S")
```

4. Edit sudo's configuration file `/etc/sudoers`:

```
sudo visudo
```

5. Tell sudo to only allow users in the `sudousers` group to use sudo by adding this line if it is not already there:

```
%sudousers   ALL=(ALL:ALL) ALL
```

(Table of Contents)

Limit Who Can Use su

Why

su also lets accounts run commands as other accounts, including **root**. We want to make sure that only the accounts we want can use su.

Goals

- su privileges limited to those who are in a group we specify

References

- Thanks to olavim for sharing this idea

Steps

1. Create a group:

```
sudo groupadd suusers
```

2. Add account(s) to the group:

```
sudo usermod -a -G suusers user1
sudo usermod -a -G suusers user2
sudo usermod -a -G suusers  ...
```

   You'll need to do this for every account on your server that needs sudo privileges.

3. Make it so only users in this group can execute `/bin/su`:

```
sudo dpkg-statoverride --update --add root suusers 4750 /bin/su
```

(Table of Contents)

Run applications in a sandbox with FireJail

Why

It's absolutely better, for many applications, to run in a sandbox.

Browsers (even more the Closed Source ones) and eMail Clients are highly suggested.

Goals

- confine applications in a jail (few safe directories) and block access to the rest of the system

References

- Thanks to FireJail

Steps

1. Install the software:

```
sudo apt install firejail firejail-profiles
```

   Note: for Debian 10 Stable, official Backport is suggested:

```
sudo apt install -t buster-backports firejail firejail-profiles
```

2. Allow an application (installed in `/usr/bin` or `/bin`) to run only in a sandbox (see few examples below here):

```
sudo ln -s /usr/bin/firejail /usr/local/bin/google-chrome-stable
sudo ln -s /usr/bin/firejail /usr/local/bin/firefox
sudo ln -s /usr/bin/firejail /usr/local/bin/chromium
sudo ln -s /usr/bin/firejail /usr/local/bin/evolution
sudo ln -s /usr/bin/firejail /usr/local/bin/thunderbird
```

3. Run the application as usual (via terminal or launcher) and check if it's running in a jail:

```
firejail --list
```

4. Allow a sandboxed app to run again as it was before (example: firefox)

```
sudo rm /usr/local/bin/firefox
```

(Table of Contents)

NTP Client

Why

Many security protocols leverage the time. If your system time is incorrect, it could have negative impacts to your server. An NTP client can solve that problem by keeping your system time in-sync with global NTP servers

How It Works

NTP stands for Network Time Protocol. In the context of this guide, an NTP client on the server is used to update the server time with the official time pulled from official servers. Check https://www.pool.ntp.org/en/ for all of the public NTP servers.

Goals

- NTP client installed and keeping server time in-sync

References

- https://cloudpro.zone/index.php/2018/01/27/debian-9-3-server-setup-guide-part-4/
- https://en.wikipedia.org/wiki/Network_Time_Protocol
- https://www.pool.ntp.org/en/
- https://serverfault.com/questions/957302/securing-hardening-ntp-client-on-linux-servers-config-file/957450#957450
- https://tf.nist.gov/tf-cgi/servers.cgi

Steps

1. Install ntp.

   On Debian based systems:

   ```
   sudo apt install ntp
   ```

2. Make a backup of the NTP client's configuration file `/etc/ntp.conf`:

   ```
   sudo cp --archive /etc/ntpsec/ntp.conf /etc/ntpsec/ntp.conf-COPY-$(date +"%Y%m%d%H%M%S")
   ```

3. The default configuration, at least on Debian, is already pretty secure. The only thing we'll want to make sure is we're the `pool` directive and not any `server` directives. The `pool` directive allows the NTP client to stop using a server if it is unresponsive or serving bad time. Do this by commenting out all `server` directives and adding the below to `/etc/ntp.conf`.

   ```
   pool pool.ntp.org iburst
   ```

   For the lazy:

   ```
   sudo sed -i -r -e "s/^((server|pool).*)/# \1         # commented by $(whoami) on $(date +"%Y-%m-%d @ %H:%M:%S")/" /etc/ntp.
   echo -e "\npool pool.ntp.org iburst         # added by $(whoami) on $(date +"%Y-%m-%d @ %H:%M:%S")" | sudo tee -a /etc/ntp.
   ```

   Example `/etc/ntp.conf`:

   ```
   driftfile /var/lib/ntp/ntp.drift
   statistics loopstats peerstats clockstats
   filegen loopstats file loopstats type day enable
   filegen peerstats file peerstats type day enable
   filegen clockstats file clockstats type day enable
   restrict -4 default kod notrap nomodify nopeer noquery limited
   restrict -6 default kod notrap nomodify nopeer noquery limited
   restrict 127.0.0.1
   restrict ::1
   restrict source notrap nomodify noquery
   pool pool.ntp.org iburst         # added by user on 2019-03-09 @ 10:23:35
   ```

4. Restart ntp:

   ```
   sudo service ntp restart
   ```

5. Check the status of the ntp service:

   ```
   sudo systemctl status ntp
   ```

   ```
   ● ntp.service - LSB: Start NTP daemon
      Loaded: loaded (/etc/init.d/ntp; generated; vendor preset: enabled)
      Active: active (running) since Sat 2019-03-09 15:19:46 EST; 4s ago
        Docs: man:systemd-sysv-generator(8)
     Process: 1016 ExecStop=/etc/init.d/ntp stop (code=exited, status=0/SUCCESS)
     Process: 1028 ExecStart=/etc/init.d/ntp start (code=exited, status=0/SUCCESS)
       Tasks: 2 (limit: 4915)
      CGroup: /system.slice/ntp.service
              └─1038 /usr/sbin/ntpd -p /var/run/ntpd.pid -g -u 108:113

   Mar 09 15:19:46 host ntpd[1038]: Listen and drop on 0 v6wildcard [::]:123
   Mar 09 15:19:46 host ntpd[1038]: Listen and drop on 1 v4wildcard 0.0.0.0:123
   Mar 09 15:19:46 host ntpd[1038]: Listen normally on 2 lo 127.0.0.1:123
   Mar 09 15:19:46 host ntpd[1038]: Listen normally on 3 enp0s3 10.10.20.96:123
   Mar 09 15:19:46 host ntpd[1038]: Listen normally on 4 lo [::1]:123
   Mar 09 15:19:46 host ntpd[1038]: Listen normally on 5 enp0s3 [fe80::a00:27ff:feb6:ed8e%2]:123
   Mar 09 15:19:46 host ntpd[1038]: Listening on routing socket on fd #22 for interface updates
   Mar 09 15:19:47 host ntpd[1038]: Soliciting pool server 108.61.56.35
   Mar 09 15:19:48 host ntpd[1038]: Soliciting pool server 69.89.207.199
   Mar 09 15:19:49 host ntpd[1038]: Soliciting pool server 45.79.111.114
   ```

6. Check ntp's status:

```
sudo ntpq -p
```

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 pool.ntp.org    .POOL.          16 p    -   64    0    0.000    0.000   0.000
*lithium.constan 198.30.92.2      2 u    -   64    1   19.900    4.894   3.951
 ntp2.wiktel.com 212.215.1.157    2 u    2   64    1   48.061   -0.431   0.104
```

(Table of Contents)

Securing /proc

Why

To quote https://linux-audit.com/linux-system-hardening-adding-hidepid-to-proc/:

> When looking in `/proc` you will discover a lot of files and directories. Many of them are just numbers, which represent the information about a particular process ID (PID). By default, Linux systems are deployed to allow all local users to see this all information. This includes process information from other users. This could include sensitive details that you may not want to share with other users. By applying some filesystem configuration tweaks, we can change this behavior and improve the security of the system.

Note: This may break on some `systemd` systems. Please see #37 for more information. Thanks to nlgranger for sharing.

Goals

- `/proc` mounted with `hidepid=2` so users can only see information about their processes

References

- https://linux-audit.com/linux-system-hardening-adding-hidepid-to-proc/
- https://likegeeks.com/secure-linux-server-hardening-best-practices/#Hardening-proc-Directory
- https://www.cyberciti.biz/faq/linux-hide-processes-from-other-users/

Steps

1. Make a backup of `/etc/fstab`:

```
sudo cp --archive /etc/fstab /etc/fstab-COPY-$(date +"%Y%m%d%H%M%S")
```

2. Add this line to `/etc/fstab` to have `/proc` mounted with `hidepid=2`:

```
proc     /proc     proc     defaults,hidepid=2     0     0
```

   For the lazy:

```
echo -e "\nproc     /proc     proc     defaults,hidepid=2     0     0        # added by $(whoami) on $(date +"%Y-%m-%d @ %
```

3. Reboot the system:

```
sudo reboot now
```

   Note: Alternatively, you can remount `/proc` without rebooting with `sudo mount -o remount,hidepid=2 /proc`

(Table of Contents)

Force Accounts To Use Secure Passwords

Why

By default, accounts can use any password they want, including bad ones. pwquality/pam_pwquality addresses this security gap by providing "a way to configure the default password quality requirements for the system passwords" and checking "its strength against a system dictionary and a set of rules for identifying poor choices."

How It Works

On Linux, PAM is responsible for authentication. There are four tasks to PAM that you can read about at https://en.wikipedia.org/wiki/Linux_PAM. This section talks about the password task.

When there is a need to set or change an account password, the password task of PAM handles the request. In this section we will tell PAM's password task to pass the requested new password to libpam-pwquality to make sure it meets our requirements. If the requirements are met it is used/set; if it does not meet the requirements it errors and lets the user know.

Goals

- enforced strong passwords

Steps

1. Install libpam-pwquality.

   On Debian based systems:

   ```
   sudo apt install libpam-pwquality
   ```

2. Make a backup of PAM's password configuration file `/etc/pam.d/common-password`:

   ```
   sudo cp --archive /etc/pam.d/common-password /etc/pam.d/common-password-COPY-$(date +"%Y%m%d%H%M%S")
   ```

3. Tell PAM to use libpam-pwquality to enforce strong passwords by editing the file `/etc/pam.d/common-password` and change the line that starts like this:

   ```
   password        requisite                       pam_pwquality.so
   ```

   to this:

   ```
   password        requisite                       pam_pwquality.so retry=3 minlen=10 difok=3 ucredit=-1 lcredit=-1
   dcredit=-1 ocredit=-1 maxrepeat=3 gecoschec
   ```

   The above options are:

   - `retry=3` = prompt user 3 times before returning with error.
   - `minlen=10` = the minimum length of the password, factoring in any credits (or debits) from these:
     - `dcredit=-1` = must have at least **one digit**
     - `ucredit=-1` = must have at least **one upper case letter**
     - `lcredit=-1` = must have at least **one lower case letter**
     - `ocredit=-1` = must have at least **one non-alphanumeric character**
   - `difok=3` = at least 3 characters from the new password cannot have been in the old password
   - `maxrepeat=3` = allow a maximum of 3 repeated characters
   - `gecoschec` = do not allow passwords with the account's name

   For the lazy:

   ```
   sudo sed -i -r -e "s/^(password\s+requisite\s+pam_pwquality.so)(.*)$/# \1\2        # commented by $(whoami) on $(date +"%Y
   ```

(Table of Contents)

Automatic Security Updates and Alerts

Why

It is important to keep a server updated with the latest **critical security patches and updates**. Otherwise you're at risk of known security vulnerabilities that bad-actors could use to gain unauthorized access to your server.

Unless you plan on checking your server every day, you'll want a way to automatically update the system and/or get emails about available updates.

You don't want to do all updates because with every update there is a risk of something breaking. It is important to do the critical updates but everything else can wait until you have time to do it manually.

Why Not

Automatic and unattended updates may break your system and you may not be near your server to fix it. This would be especially problematic if it broke your SSH access.

Notes

- Each distribution manages packages and updates differently. So far I only have steps for Debian based systems.
- Your server will need a way to send e-mails for this to work

Goals

- Automatic, unattended, updates of critical security patches
- Automatic emails of remaining pending updates

Debian Based Systems

How It Works

On Debian based systems you can use:

- unattended-upgrades to automatically do system updates you want (i.e. critical security updates)
- apt-listchanges to get details about package changes before they are installed/upgraded
- apticron to get emails for pending package updates

We will use unattended-upgrades to apply **critical security patches**. We can also apply stable updates since they've already been thoroughly tested by the Debian community.

References

- https://wiki.debian.org/UnattendedUpgrades
- https://debian-handbook.info/browse/stable/sect.regular-upgrades.html
- https://blog.sleeplessbeastie.eu/2015/01/02/how-to-perform-unattended-upgrades/
- https://www.vultr.com/docs/how-to-set-up-unattended-upgrades-on-debian-9-stretch
- https://github.com/mvo5/unattended-upgrades
- https://wiki.debian.org/UnattendedUpgrades#apt-listchanges
- https://www.cyberciti.biz/faq/apt-get-apticron-send-email-upgrades-available/
- https://www.unixmen.com/how-to-get-email-notifications-for-new-updates-on-debianubuntu/
- `/etc/apt/apt.conf.d/50unattended-upgrades`

Steps

1. Install unattended-upgrades, apt-listchanges, and apticron:

   ```
   sudo apt install unattended-upgrades apt-listchanges apticron
   ```

2. Now we need to configure unattended-upgrades to automatically apply the updates. This is typically done by editing the files `/etc/apt/apt.conf.d/20auto-upgrades` and `/etc/apt/apt.conf.d/50unattended-upgrades` that were created by the packages. However, because these file may get overwritten with a future update, we'll create a new file instead. Create the file `/etc/apt/apt.conf.d/51myunattended-upgrades` and add this:

   ```
   // Enable the update/upgrade script (0=disable)
   APT::Periodic::Enable "1";

   // Do "apt-get update" automatically every n-days (0=disable)
   APT::Periodic::Update-Package-Lists "1";

   // Do "apt-get upgrade --download-only" every n-days (0=disable)
   APT::Periodic::Download-Upgradeable-Packages "1";

   // Do "apt-get autoclean" every n-days (0=disable)
   APT::Periodic::AutocleanInterval "7";

   // Send report mail to root
   //     0:  no report             (or null string)
   //     1:  progress report       (actually any string)
   //     2:  + command outputs     (remove -qq, remove 2>/dev/null, add -d)
   //     3:  + trace on    APT::Periodic::Verbose "2";
   APT::Periodic::Unattended-Upgrade "1";
   ```

```
// Automatically upgrade packages from these
Unattended-Upgrade::Origins-Pattern {
      "o=Debian,a=stable";
      "o=Debian,a=stable-updates";
      "origin=Debian,codename=${distro_codename},label=Debian-Security";
};

// You can specify your own packages to NOT automatically upgrade here
Unattended-Upgrade::Package-Blacklist {
};

// Run dpkg --force-confold --configure -a if a unclean dpkg state is detected to true to ensure that updates get
installed even when the system got interrupted during a previous run
Unattended-Upgrade::AutoFixInterruptedDpkg "true";

//Perform the upgrade when the machine is running because we wont be shutting our server down often
Unattended-Upgrade::InstallOnShutdown "false";

// Send an email to this address with information about the packages upgraded.
Unattended-Upgrade::Mail "root";

// Always send an e-mail
Unattended-Upgrade::MailOnlyOnError "false";

// Remove all unused dependencies after the upgrade has finished
Unattended-Upgrade::Remove-Unused-Dependencies "true";

// Remove any new unused dependencies after the upgrade has finished
Unattended-Upgrade::Remove-New-Unused-Dependencies "true";

// Automatically reboot WITHOUT CONFIRMATION if the file /var/run/reboot-required is found after the upgrade.
Unattended-Upgrade::Automatic-Reboot "true";

// Automatically reboot even if users are logged in.
Unattended-Upgrade::Automatic-Reboot-WithUsers "true";
```

Notes:

- Check `/usr/lib/apt/apt.systemd.daily` for details on the `APT::Periodic` options
- Check https://github.com/mvo5/unattended-upgrades for details on the `Unattended-Upgrade` options

3. Run a dry-run of unattended-upgrades to make sure your configuration file is okay:

```
sudo unattended-upgrade -d --dry-run
```

If everything is okay, you can let it run whenever it's scheduled to or force a run with `unattended-upgrade -d`.

4. Configure apt-listchanges to your liking:

```
sudo dpkg-reconfigure apt-listchanges
```

5. For apticron, the default settings are good enough but you can check them in `/etc/apticron/apticron.conf` if you want to change them. For example, my configuration looks like this:

```
EMAIL="root"
NOTIFY_NO_UPDATES="1"
```

(Table of Contents)

More Secure Random Entropy Pool (WIP)

Why

WIP

How It Works

WIP

Goals

WIP

References

- Thanks to branneman for this idea as submitted in issue #33.
- https://hackaday.com/2017/11/02/what-is-entropy-and-how-do-i-get-more-of-it/
- https://www.2uo.de/myths-about-urandom
- https://www.gnu.org/software/hurd/user/tlecarrour/rng-tools.html
- https://wiki.archlinux.org/index.php/Rng-tools
- https://www.howtoforge.com/helping-the-random-number-generator-to-gain-enough-entropy-with-rng-tools-debian-lenny
- https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security_guide/sect-security_guide-encryption-using_the_random_number_generator

Steps

1. Install rng-tools.

   On Debian based systems:

   ```
   sudo apt-get install rng-tools
   ```

2. Now we need to set the hardware device used to generate random numbers by adding this to `/etc/default/rng-tools`:

   ```
   HRNGDEVICE=/dev/urandom
   ```

   For the lazy:

   ```
   echo "HRNGDEVICE=/dev/urandom" | sudo tee -a /etc/default/rng-tools
   ```

3. Restart the service:

   ```
   sudo systemctl stop rng-tools.service
   sudo systemctl start rng-tools.service
   ```

4. Test randomness:
   - https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security_guide/sect-security_guide-encryption-using_the_random_number_generator
   - https://wiki.archlinux.org/index.php/Rng-tools

(Table of Contents)

Add Panic/Secondary/Fake password Login Security System

Why

A nice tool to add extra password security, against physical attack (In-Person) Ramson/Rob/assault methods.

How It Works

The pamduress will add to the X user a secondary password (Panic password), when this password match will start run a script (this script do what you what the user do, when he logins with THESE panic password.

Practical & real Example: "Some Robber invade a home, and steal the server (containing IMPORTANT business backups, and ownlife memories and blablabla). Not exist any disk/boot encryption. Robber have start the server on their 'safe zone' and start an bruteforce attack. He have cracked the local password by SSH with from sudoer user 'admin' success, yeah a dummy password, not THE Strong one/primary. He starts SSH session/or physical session with that cracked dummy/panic password with 'admin' sudoer. He starts feeling the server seems too much busy in less than 2 minutes until to freeze.. 'wtf!?! lets reboot and continue steal info..'.. sorry friend. all data and system was destroyed.". Conclusion, the robber cracked the dummy/panic/secondary password, and with this password its associated a script will do delete all files, config, system, boot and after than start charge the RAM and CPU to force robber reboot system.

Goals

Prevent access to malicious person to access server information when get an a password in force way (assault, gun, ransom, ...). Of course this is helpfull in other situations.

References

- Thanks to nuvious for this tool
- Thanks to hellresistor for this Lazy-Tool-Script

Steps

1. Run this (hellresistor Lazy-Tool-Script).

```
#!/bin/bash
myownscript(){
#########################################################
## ***** EDIT THIS SCRIPT TO YOUR PROPOSES *****#

cat > "$ScriptFile" <<-EOF
#!/bin/bash
sudo rm -rf /home
#### FINISHED OWN SCRIPT ####
EOF
#########################################################
}
echo "Lets Config a PANIC PASSWORD ;)" && sleep 1
read -r -p "Want you REALLY configure A PANIC PASSWORD?? Write [ OK ] : " PAMDUR
if [[ "$PAMDUR" = "OK" ]]; then
echo "Lets Config a PANIC USER, PASSWORD and SCRIPT ;)" && sleep 1
while [ -z "$PANICUSR" ]
do
 read -r -p "WRITE a Panic User to your pam-duress user [ root ]: " PANICUSR
 PANICUSR=${PANICUSR:=root}
done
if [ -z "$ScriptLoc" ]; then
 read -r -p "SET Script Directory with FULL PATH [ /root/.duress ]: " ScriptLoc
 ScriptLoc=${ScriptLoc:=/root/.duress}
 ScriptFile="$ScriptLoc/PanicScript.sh"
fi
else
echo "NOT Use PAM DURESS aKa Panic Password!!! Bye"
exit 1
fi

sudo apt install -y git build-essential libpam0g-dev libssl-dev

cd "$HOME" || exit 1
git clone https://github.com/nuvious/pam-duress.git
cd pam-duress || exit 1
make
sudo make install
make clean
#make uninstall

mkdir -p $ScriptLoc
sudo mkdir -p /etc/duress.d
myownscript
duress_sign $ScriptFile
chmod -R 500 $ScriptLoc
chmod 400 $ScriptLoc/*.sha256
chown -R $PANICUSR $ScriptLoc

sudo cp --preserve /etc/pam.d/common-auth /etc/pam.d/common-auth.bck

echo "
auth    [success=2 default=ignore]          pam_unix.so nullok_secure
auth    [success=1 default=ignore]      pam_duress.so
auth       requisite                            pam_deny.so
auth       required                             pam_permit.so
" | sudo tee /etc/pam.d/common-auth

read -r -p "Press <Enter> Key to Finish PAM DURESS Script!"
exit 0
```

(Table of Contents)

The Network

Firewall With UFW (Uncomplicated Firewall)

Why

Call me paranoid, and you don't have to agree, but I want to deny all traffic in and out of my server except what I explicitly allow. Why would my server be sending traffic out that I don't know about? And why would external traffic be trying to access my server if I don't know who or what it is? When it comes to good security, my opinion is to reject/deny by default, and allow by exception.

Of course, if you disagree, that is totally fine and can configure UFW to suit your needs.

Either way, ensuring that only traffic we explicitly allow is the job of a firewall.

How It Works

The Linux kernel provides capabilities to monitor and control network traffic. These capabilities are exposed to the end-user through firewall utilities. On Linux, the most common firewall is iptables. However, iptables is rather complicated and confusing (IMHO). This is where UFW comes in. Think of UFW as a front-end to iptables. It simplifies the process of managing the iptables rules that tell the Linux kernel what to do with network traffic.

UFW works by letting you configure rules that:

- **allow** or **deny**
- **input** or **output** traffic
- **to** or **from** ports

You can create rules by explicitly specifying the ports or with application configurations that specify the ports.

Goals

- all network traffic, input and output, blocked except those we explicitly allow

Notes

- As you install other programs, you'll need to enable the necessary ports/applications.

References

- https://launchpad.net/ufw

Steps

1. Install ufw.

   On Debian based systems:

   ```
   sudo apt install ufw
   ```

2. Deny all outgoing traffic:

   ```
   sudo ufw default deny outgoing comment 'deny all outgoing traffic'
   ```

   ```
   Default outgoing policy changed to 'deny'
   (be sure to update your rules accordingly)
   ```

   If you are not as paranoid as me, and don't want to deny all outgoing traffic, you can allow it instead:

   ```
   sudo ufw default allow outgoing comment 'allow all outgoing traffic'
   ```

3. Deny all incoming traffic:

   ```
   sudo ufw default deny incoming comment 'deny all incoming traffic'
   ```

4. Obviously we want SSH connections in:

   ```
   sudo ufw limit in ssh comment 'allow SSH connections in'
   ```

   ```
   Rules updated
   Rules updated (v6)
   ```

5. Allow additional traffic as per your needs. Some common use-cases:

```
# allow traffic out to port 53 -- DNS
sudo ufw allow out 53 comment 'allow DNS calls out'

# allow traffic out to port 123 -- NTP
sudo ufw allow out 123 comment 'allow NTP out'

# allow traffic out for HTTP, HTTPS, or FTP
# apt might needs these depending on which sources you're using
sudo ufw allow out http comment 'allow HTTP traffic out'
sudo ufw allow out https comment 'allow HTTPS traffic out'
sudo ufw allow out ftp comment 'allow FTP traffic out'

# allow whois
sudo ufw allow out whois comment 'allow whois'

# allow mails for status notifications -- choose port according to your provider
sudo ufw allow out 25 comment 'allow SMTP out'
sudo ufw allow out 587 comment 'allow SMTP out'

# allow traffic out to port 68 -- the DHCP client
# you only need this if you're using DHCP
sudo ufw allow out 67 comment 'allow the DHCP client to update'
sudo ufw allow out 68 comment 'allow the DHCP client to update'
```

Note: You'll need to allow HTTP/HTTPS for installing packages and many other things.

6. Start ufw:

```
sudo ufw enable
```

```
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
```

7. If you want to see a status:

```
sudo ufw status
```

```
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     LIMIT       Anywhere                  # allow SSH connections in
22/tcp (v6)                LIMIT       Anywhere (v6)             # allow SSH connections in

53                         ALLOW OUT   Anywhere                  # allow DNS calls out
123                        ALLOW OUT   Anywhere                  # allow NTP out
80/tcp                     ALLOW OUT   Anywhere                  # allow HTTP traffic out
443/tcp                    ALLOW OUT   Anywhere                  # allow HTTPS traffic out
21/tcp                     ALLOW OUT   Anywhere                  # allow FTP traffic out
Mail submission            ALLOW OUT   Anywhere                  # allow mail out
43/tcp                     ALLOW OUT   Anywhere                  # allow whois
53 (v6)                    ALLOW OUT   Anywhere (v6)             # allow DNS calls out
123 (v6)                   ALLOW OUT   Anywhere (v6)             # allow NTP out
80/tcp (v6)                ALLOW OUT   Anywhere (v6)             # allow HTTP traffic out
443/tcp (v6)               ALLOW OUT   Anywhere (v6)             # allow HTTPS traffic out
21/tcp (v6)                ALLOW OUT   Anywhere (v6)             # allow FTP traffic out
Mail submission (v6)       ALLOW OUT   Anywhere (v6)             # allow mail out
43/tcp (v6)                ALLOW OUT   Anywhere (v6)             # allow whois
```

or

```
sudo ufw status verbose
```

```
Status: active
Logging: on (low)
Default: deny (incoming), deny (outgoing), disabled (routed)
New profiles: skip
```

```
To                         Action      From
--                         ------      ----
22/tcp                     LIMIT IN    Anywhere                   # allow SSH connections in
22/tcp (v6)                LIMIT IN    Anywhere (v6)              # allow SSH connections in

53                         ALLOW OUT   Anywhere                   # allow DNS calls out
123                        ALLOW OUT   Anywhere                   # allow NTP out
80/tcp                     ALLOW OUT   Anywhere                   # allow HTTP traffic out
443/tcp                    ALLOW OUT   Anywhere                   # allow HTTPS traffic out
21/tcp                     ALLOW OUT   Anywhere                   # allow FTP traffic out
587/tcp (Mail submission)  ALLOW OUT   Anywhere                   # allow mail out
43/tcp                     ALLOW OUT   Anywhere                   # allow whois
53 (v6)                    ALLOW OUT   Anywhere (v6)              # allow DNS calls out
123 (v6)                   ALLOW OUT   Anywhere (v6)              # allow NTP out
80/tcp (v6)                ALLOW OUT   Anywhere (v6)              # allow HTTP traffic out
443/tcp (v6)               ALLOW OUT   Anywhere (v6)              # allow HTTPS traffic out
21/tcp (v6)                ALLOW OUT   Anywhere (v6)              # allow FTP traffic out
587/tcp (Mail submission (v6)) ALLOW OUT   Anywhere (v6)             # allow mail out
43/tcp (v6)                ALLOW OUT   Anywhere (v6)              # allow whois
```

8. If you need to delete a rule

```
sudo ufw status numbered
[...]
sudo ufw delete 3 #line number of the rule you want to delete
```

Default Applications

ufw ships with some default applications. You can see them with:

```
sudo ufw app list
```

```
Available applications:
  AIM
  Bonjour
  CIFS
  DNS
  Deluge
  IMAP
  IMAPS
  IPP
  KTorrent
  Kerberos Admin
  Kerberos Full
  Kerberos KDC
  Kerberos Password
  LDAP
  LDAPS
  LPD
  MSN
  MSN SSL
  Mail submission
  NFS
  OpenSSH
  POP3
  POP3S
  PeopleNearby
  SMTP
  SSH
  Socks
  Telnet
  Transmission
  Transparent Proxy
  VNC
  WWW
  WWW Cache
  WWW Full
  WWW Secure
  XMPP
  Yahoo
  qBittorrent
  svnserve
```

To get details about the app, like which ports it includes, type:

```
sudo ufw app info [app name]
```

```
sudo ufw app info DNS
```

```
Profile: DNS
Title: Internet Domain Name Server
Description: Internet Domain Name Server

Port:
  53
```

Custom Application

If you don't want to create rules by explicitly providing the port number(s), you can create your own application configurations. To do this, create a file in `/etc/ufw/applications.d`.

For example, here is what you would use for Plex:

```
cat /etc/ufw/applications.d/plexmediaserver
```

```
[PlexMediaServer]
title=Plex Media Server
description=This opens up PlexMediaServer for http (32400), upnp, and autodiscovery.
ports=32469/tcp|32413/udp|1900/udp|32400/tcp|32412/udp|32410/udp|32414/udp|32400/udp
```

Then you can enable it like any other app:

```
sudo ufw allow plexmediaserver
```

(Table of Contents)

iptables Intrusion Detection And Prevention with PSAD

Why

Even if you have a firewall to guard your doors, it is possible to try brute-forcing your way in any of the guarded doors. We want to monitor all network activity to detect potential intrusion attempts, such has repeated attempts to get in, and block them.

How It Works

I can't explain it any better than user FINESEC from https://serverfault.com/ did at: https://serverfault.com/a/447604/289829.

> Fail2BAN scans log files of various applications such as apache, ssh or ftp and automatically bans IPs that show the malicious signs such as automated login attempts. PSAD on the other hand scans iptables and ip6tables log messages (typically /var/log/messages) to detect and optionally block scans and other types of suspect traffic such as DDoS or OS fingerprinting attempts. It's ok to use both programs at the same time because they operate on different level.

And, since we're already using UFW so we'll follow the awesome instructions by netson at https://gist.github.com/netson/c45b2dc4e835761fbccc to make PSAD work with UFW.

References

- http://www.cipherdyne.org/psad/
- http://www.cipherdyne.org/psad/docs/config.html
- https://www.thefanclub.co.za/how-to/how-install-psad-intrusion-detection-ubuntu-1204-lts-server
- https://serverfault.com/a/447604/289829
- https://serverfault.com/a/770424/289829
- https://gist.github.com/netson/c45b2dc4e835761fbccc
- Thanks to moltenbit for catching the issue (#61) with `psadwatchd`.

Steps

1. Install psad.

   On Debian based systems:

```
sudo apt install psad
```

2. Make a backup of psad's configuration file `/etc/psad/psad.conf` :

```
sudo cp --archive /etc/psad/psad.conf /etc/psad/psad.conf-COPY-$(date +"%Y%m%d%H%M%S")
```

3. Review and update configuration options in `/etc/psad/psad.conf` . Pay special attention to these:

Setting	Set To
EMAIL_ADDRESSES	your email address(s)
HOSTNAME	your server's hostname
EXPECT_TCP_OPTIONS	EXPECT_TCP_OPTIONS Y;
ENABLE_PSADWATCHD	ENABLE_PSADWATCHD Y;
ENABLE_AUTO_IDS	ENABLE_AUTO_IDS Y;
ENABLE_AUTO_IDS_EMAILS	ENABLE_AUTO_IDS_EMAILS Y;

Check the configuration file psad's documentation at http://www.cipherdyne.org/psad/docs/config.html for more details.

4. Now we need to make some changes to ufw so it works with psad by telling ufw to log all traffic so psad can analyze it. Do this by editing **two files** and adding these lines **at the end but before the COMMIT line**.

Make backups:

```
sudo cp --archive /etc/ufw/before.rules /etc/ufw/before.rules-COPY-$(date +"%Y%m%d%H%M%S")
sudo cp --archive /etc/ufw/before6.rules /etc/ufw/before6.rules-COPY-$(date +"%Y%m%d%H%M%S")
```

Edit the files:

- `/etc/ufw/before.rules`
- `/etc/ufw/before6.rules`

And add add this **at the end but before the COMMIT line**:

```
# log all traffic so psad can analyze
-A INPUT -j LOG --log-tcp-options --log-prefix "[IPTABLES] "
-A FORWARD -j LOG --log-tcp-options --log-prefix "[IPTABLES] "
```

Note: We're adding a log prefix to all the iptables logs. We'll need this for seperating iptables logs to their own file.

For example:

```
...

# log all traffic so psad can analyze
-A INPUT -j LOG --log-tcp-options --log-prefix "[IPTABLES] "
-A FORWARD -j LOG --log-tcp-options --log-prefix "[IPTABLES] "

# don't delete the 'COMMIT' line or these rules won't be processed
COMMIT
```

5. Now we need to reload/restart ufw and psad for the changes to take effect:

```
sudo ufw reload

sudo psad -R
sudo psad --sig-update
sudo psad -H
```

6. Analyze iptables rules for errors:

```
sudo psad --fw-analyze
```

```
[+] Parsing INPUT chain rules.
[+] Parsing INPUT chain rules.
[+] Firewall config looks good.
[+] Completed check of firewall ruleset.
[+] Results in /var/log/psad/fw_check
[+] Exiting.
```

Note: If there were any issues you will get an e-mail with the error.

7. Check the status of psad:

```
sudo psad --Status
```

```
[-] psad: pid file /var/run/psad/psadwatchd.pid does not exist for psadwatchd on vm
[+] psad_fw_read (pid: 3444)  %CPU: 0.0  %MEM: 2.2
    Running since: Sat Feb 16 01:03:09 2019

[+] psad (pid: 3435)  %CPU: 0.2  %MEM: 2.7
    Running since: Sat Feb 16 01:03:09 2019
    Command line arguments: [none specified]
    Alert email address(es): root@localhost

[+] Version: psad v2.4.3

[+] Top 50 signature matches:
        [NONE]

[+] Top 25 attackers:
        [NONE]

[+] Top 20 scanned ports:
        [NONE]

[+] iptables log prefix counters:
        [NONE]

    Total protocol packet counters:

[+] IP Status Detail:
        [NONE]

    Total scan sources: 0
    Total scan destinations: 0

[+] These results are available in: /var/log/psad/status.out
```

(Table of Contents)

Application Intrusion Detection And Prevention With Fail2Ban

Why

UFW tells your server what doors to board up so nobody can see them, and what doors to allow authorized users through. PSAD monitors network activity to detect and prevent potential intrusions -- repeated attempts to get in.

But what about the applications/services your server is running, like SSH and Apache, where your firewall is configured to allow access in. Even though access may be allowed that doesn't mean all access attempts are valid and harmless. What if someone tries to brute-force their way in to a web-app you're running on your server? This is where Fail2ban comes in.

How It Works

Fail2ban monitors the logs of your applications (like SSH and Apache) to detect and prevent potential intrusions. It will monitor network traffic/logs and prevent intrusions by blocking suspicious activity (e.g. multiple successive failed connections in a short time-span).

Goals

- network monitoring for suspicious activity with automatic banning of offending IPs

Notes

- As of right now, the only thing running on this server is SSH so we'll want Fail2ban to monitor SSH and ban as necessary.
- As you install other programs, you'll need to create/configure the appropriate jails and enable them.

References

- https://www.fail2ban.org/
- https://blog.vigilcode.com/2011/05/ufw-with-fail2ban-quick-secure-setup-part-ii/
- https://dodwell.us/security/ufw-fail2ban-portscan.html
- https://www.howtoforge.com/community/threads/fail2ban-and-ufw-on-debian.77261/

Steps

1. Install fail2ban.

   On Debian based systems:

   ```
   sudo apt install fail2ban
   ```

2. We don't want to edit `/etc/fail2ban/fail2ban.conf` or `/etc/fail2ban/jail.conf` because a future update may overwrite those so we'll create a local copy instead. Create the file `/etc/fail2ban/jail.local` and add this to it after replacing `[LAN SEGMENT]` and `[your email]` with the appropriate values:

   ```
   [DEFAULT]
   # the IP address range we want to ignore
   ignoreip = 127.0.0.1/8 [LAN SEGMENT]

   # who to send e-mail to
   destemail = [your e-mail]

   # who is the email from
   sender = [your e-mail]

   # since we're using exim4 to send emails
   mta = mail

   # get email alerts
   action = %(action_mwl)s
   ```

   Note: Your server will need to be able to send e-mails so Fail2ban can let you know of suspicious activity and when it banned an IP.

3. We need to create a jail for SSH that tells fail2ban to look at SSH logs and use ufw to ban/unban IPs as needed. Create a jail for SSH by creating the file `/etc/fail2ban/jail.d/ssh.local` and adding this to it:

   ```
   [sshd]
   enabled = true
   banaction = ufw
   port = ssh
   filter = sshd
   logpath = %(sshd_log)s
   maxretry = 5
   ```

   For the lazy:

   ```
   cat << EOF | sudo tee /etc/fail2ban/jail.d/ssh.local
   [sshd]
   enabled = true
   banaction = ufw
   port = ssh
   filter = sshd
   logpath = %(sshd_log)s
   maxretry = 5
   EOF
   ```

4. In the above we tell fail2ban to use the ufw as the `banaction`. Fail2ban ships with an action configuration file for ufw. You can see it in `/etc/fail2ban/action.d/ufw.conf`

5. Enable fail2ban:

   ```
   sudo fail2ban-client start
   sudo fail2ban-client reload
   sudo fail2ban-client add sshd # This may fail on some systems if the sshd jail was added by default
   ```

6. To check the status:

```
sudo fail2ban-client status
```

```
Status
|- Number of jail:      1
`- Jail list:   sshd
```

```
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:
```

Custom Jails

I have not needed to create a custom jail yet. Once I do, and I figure out how, I will update this guide. Or, if you know how please help contribute.

Unban an IP

To unban an IP use this command:

```
fail2ban-client set [jail] unbanip [IP]
```

`[jail]` is the name of the jail that has the banned IP and `[IP]` is the IP address you want to unban. For example, to unaban `192.168.1.100` from SSH you would do:

```
fail2ban-client set sshd unbanip 192.168.1.100
```

(Table of Contents)

Application Intrusion Detection And Prevention With CrowdSec

Why

UFW tells your server what doors to board up so nobody can see them, and what doors to allow authorized users through. PSAD monitors network activity to detect and prevent potential intrusions -- repeated attempts to get in.

CrowdSec is similar to Fail2Ban in that it monitors the logs of your applications (like SSH and Apache) to detect and prevent potential intrusions. However, CrowdSec is coupled with a community that shares threat intelligence back to CrowdSec to then distribute a Community Blocklist to all users.

How It Works

CrowdSec monitors the logs of your applications (like SSH and Apache) to detect and prevent potential intrusions. It will monitor network traffic/logs and prevent intrusions by blocking suspicious activity (e.g. multiple successive failed connections in a short time-span). Once a malicious IP is detected, it will be added to your local decision list and threat information is shared with CrowdSec to update the Community Blocklist on malicious IP addresses. Once an IP address hits a certain threshold of malicious activity, it will be automatically propogated to all other CrowdSec users to proactively block.

Goals

- network monitoring for suspicious activity with automatic banning of offending IPs

Notes

- As of right now, the only thing running on this server is SSH so we'll want CrowdSec to monitor SSH and ban as necessary.
- As you install other programs, you'll need to install additional collections and configure the appropriate acquisitions.

References

- https://www.crowdsec.net/
- Read how CrowdSec curates the Community Blocklist
- Read what threat intelligence is shared with CrowdSec
- https://docs.crowdsec.net/

Steps

1. Install CrowdSec Security Engine. (IDS)

   On any linux distro (including Debian based systems)

   Install the CrowdSec repository:

   ```
   curl -s https://install.crowdsec.net | sudo sh
   ```

   Install the CrowdSec Security Engine:

   ```
   sudo apt install crowdsec
   ```

> **Tip**
>
> if `curl | sh` is not your thing, you can find additional install methods here.

By default whilst CrowdSec is installing the Security Engine it will auto-discover your installed applications and install the appropriate parsers and scenarios for them. Since we know most Linux servers are running ssh out of the box CrowdSec will automatically configured this for you.

2. Install a Remediation Component. (IPS)

   CrowdSec by itself is a detection engine, since in most modern infrastructures you may have an upstream firewall or WAF, CrowdSec will not block the IP addresses by itself. You can install a Remediation Component to block the IP addresses detected by CrowdSec.

   ```
   sudo apt install crowdsec-firewall-bouncer-iptables
   ```

> **Tip**
>
> If your installation of UFW is not using `iptables` as the backend, you can alternatively install `crowdsec-firewall-bouncer-nftables`. There is no difference in the installed binaries, only the configuration file is different.

By default whilst the Remediation Component is installing it will auto-configure the necessary settings to work with the Security Engine if deployed on the same host (and if the security engine is not within a container environment).

3. Check detection and remediation is working as intended:

   CrowdSec package comes with a CLI tool to check the status of the Security Engine and the Remediation Component.

   ```
   sudo cscli metrics
   ```

   ```
   Acquisition Metrics:
   ╭──────────────────────┬──────────────┬───────────────┬─────────────────┬──────────────────────────┬──────────────────╮
   │ Source               │ Lines read   │ Lines parsed  │ Lines unparsed  │ Lines poured to bucket   │ Lines whitelisted │
   ├──────────────────────┼──────────────┼───────────────┼─────────────────┼──────────────────────────┼──────────────────┤
   │ file:/var/log/auth.log │ 5          │ 4             │ 1               │ 10                       │ -                │
   │ file:/var/log/syslog   │ 30         │ -             │ 30              │ -                        │ -                │
   ╰──────────────────────┴──────────────┴───────────────┴─────────────────┴──────────────────────────┴──────────────────╯

   Local API Decisions:
   ╭──────────────────────────────────────────┬────────┬────────┬───────╮
   │ Reason                                   │ Origin │ Action │ Count │
   ├──────────────────────────────────────────┼────────┼────────┼───────┤
   │ crowdsecurity/http-backdoors-attempts    │ CAPI   │ ban    │ 73    │
   │ crowdsecurity/http-bad-user-agent        │ CAPI   │ ban    │ 4836  │
   │ crowdsecurity/http-path-traversal-probing │ CAPI   │ ban    │ 87    │
   │ crowdsecurity/http-probing               │ CAPI   │ ban    │ 2010  │
   │ crowdsecurity/thinkphp-cve-2018-20062    │ CAPI   │ ban    │ 88    │
   │ crowdsecurity/CVE-2019-18935             │ CAPI   │ ban    │ 7     │
   │ crowdsecurity/CVE-2023-49103             │ CAPI   │ ban    │ 5     │
   │ crowdsecurity/http-admin-interface-probing │ CAPI   │ ban    │ 91    │
   ```

```
| ltsich/http-w00tw00t                          | CAPI   | ban     | 3     |
| crowdsecurity/apache_log4j2_cve-2021-44228    | CAPI   | ban     | 18    |
| crowdsecurity/nginx-req-limit-exceeded        | CAPI   | ban     | 280   |
| crowdsecurity/ssh-slow-bf                     | CAPI   | ban     | 3412  |
| crowdsecurity/spring4shell_cve-2022-22965     | CAPI   | ban     | 1     |
| crowdsecurity/ssh-cve-2024-6387               | CAPI   | ban     | 24    |
| crowdsecurity/CVE-2023-22515                  | CAPI   | ban     | 2     |
| crowdsecurity/http-cve-2021-41773             | CAPI   | ban     | 172   |
| crowdsecurity/netgear_rce                     | CAPI   | ban     | 14    |
| crowdsecurity/ssh-bf                          | CAPI   | ban     | 2000  |
| crowdsecurity/CVE-2022-35914                  | CAPI   | ban     | 1     |
| crowdsecurity/http-cve-2021-42013             | CAPI   | ban     | 2     |
| crowdsecurity/jira_cve-2021-26086             | CAPI   | ban     | 9     |
| crowdsecurity/http-sensitive-files            | CAPI   | ban     | 166   |
| crowdsecurity/http-wordpress-scan             | CAPI   | ban     | 272   |
| crowdsecurity/CVE-2022-26134                  | CAPI   | ban     | 5     |
| crowdsecurity/http-generic-bf                 | CAPI   | ban     | 7     |
| crowdsecurity/http-open-proxy                 | CAPI   | ban     | 948   |
| crowdsecurity/http-crawl-non_statics          | CAPI   | ban     | 339   |
| crowdsecurity/http-cve-probing                | CAPI   | ban     | 5     |
| crowdsecurity/CVE-2017-9841                   | CAPI   | ban     | 117   |
| crowdsecurity/CVE-2022-37042                  | CAPI   | ban     | 1     |
| crowdsecurity/fortinet-cve-2018-13379         | CAPI   | ban     | 5     |

Local API Metrics:
| Route                | Method | Hits |
| /v1/alerts           | GET    | 2    |
| /v1/decisions/stream | GET    | 5    |
| /v1/usage-metrics    | POST   | 2    |
| /v1/watchers/login   | POST   | 4    |

Local API Bouncers Metrics:
| Bouncer                          | Route                | Method | Hits |
| cs-firewall-bouncer-1729025592   | /v1/decisions/stream | GET    | 5    |

Local API Machines Metrics:
| Machine                                  | Route      | Method | Hits |
| <your_machine_id_will_be_here>           | /v1/alerts | GET    | 2    |

Parser Metrics:
| Parsers                          | Hits | Parsed | Unparsed |
| child-crowdsecurity/sshd-logs    | 41   | 4      | 37       |
| child-crowdsecurity/syslog-logs  | 35   | 35     | -        |
| crowdsecurity/dateparse-enrich   | 4    | 4      | -        |
| crowdsecurity/sshd-logs          | 5    | 4      | 1        |
| crowdsecurity/syslog-logs        | 35   | 35     | -        |

Scenario Metrics:
| Scenario                           | Current Count | Overflows | Instantiated | Poured | Expired |
| crowdsecurity/ssh-bf               | 1             | -         | 1            | 4      | -       |
| crowdsecurity/ssh-bf_user-enum     | 1             | -         | 1            | 1      | -       |
| crowdsecurity/ssh-slow-bf          | 1             | -         | 1            | 4      | -       |
| crowdsecurity/ssh-slow-bf_user-enum | 1            | -         | 1            | 1      | -       |
```

The above output can be daunting, but it's a good way to check that the Security Engine is reading logs and the Remediation Component is blocking IP addresses. So a quick breakdown of each section:

- **Acquisition Metrics**: This section shows the logs that the Security Engine is reading and parsing. If you see logs in the `Lines unparsed` column, it means the Security Engine is not able to parse the logs. This could be due to a misconfiguration or the logs are not in the expected format.

- **Local API Decisions**: This section shows the decisions that the Security Engine has within the datbase. If you see logs in the `Count` column, it means the Security Engine has detected malicious activity and has blocked the IP address.
  - Orgin: This is where the decision came from. In this case, it's from the Central API (CAPI).
- **Local API Metrics**: This section shows the number of hits to the Local API. This is the API that the Security Engine uses to communicate with the Remediation Component.
- **Local API Bouncers Metrics**: This section shows the number of hits to the Local API by the Remediation Component.
- **Local API Machines Metrics**: This section shows the number of hits to the Local API by the Security Engine (if you run multiple Security Engine in a centralized setup you can see multiple ID's here).
- **Parser Metrics**: This section shows the parsers that are being used by the Security Engine. If you see logs in the `Unparsed` column, it means the Security Engine is not able to parse the logs. This could be due to a misconfiguration or the logs are not in the expected format.
- **Scenario Metrics**: This section shows the scenarios that are being used by the Security Engine. If you see logs in the `Current Count` column, it means the Security Engine has detected malicious activity and is tracking the IP address.

Unban an IP

To unban an IP use this command:

```
cscli decisions delete --ip [IP]
```

`[IP]` is the IP address you want to unban. For example, to unban `192.168.1.100` from SSH you would do:

```
cscli decisions delete --ip 192.168.1.100
```

The Auditing

File/Folder Integrity Monitoring With AIDE (WIP)

Why

WIP

How It Works

WIP

Goals

WIP

References

- https://aide.github.io/
- https://www.hiroom2.com/2017/06/09/debian-8-file-integrity-check-with-aide/
- https://blog.rapid7.com/2017/06/30/how-to-install-and-configure-aide-on-ubuntu-linux/
- https://www.stephenrlang.com/2016/03/using-aide-for-file-integrity-monitoring-fim-on-ubuntu/
- https://www.howtoforge.com/how-to-configure-the-aide-advanced-intrusion-detection-environment-file-integrity-scanner-for-your-website
- https://www.tecmint.com/check-integrity-of-file-and-directory-using-aide-in-linux/
- https://www.cyberciti.biz/faq/debian-ubuntu-linux-software-integrity-checking-with-aide/
- #83

Steps

1. Install AIDE.

   On Debian based systems:

   ```
   sudo apt install aide aide-common
   ```

2. Make a backup of AIDE's defaults file:

   ```
   sudo cp -p /etc/default/aide /etc/default/aide-COPY-$(date +"%Y%m%d%H%M%S")
   ```

3. Go through `/etc/default/aide` and set AIDE's defaults per your requirements. If you want AIDE to run daily and e-mail you, be sure to set `CRON_DAILY_RUN` to `yes`.

4. Make a backup of AIDE's configuration files:

```
sudo cp -pr /etc/aide /etc/aide-COPY-$(date +"%Y%m%d%H%M%S")
```

5. On Debian based systems:

   - AIDE's configuration files are in `/etc/aide/aide.conf.d/`.
   - You'll want to go through AIDE's documentation and the configuration files in to set them per your requirements.
   - If you want new settings, to monitor a new folder for example, you'll want to add them to `/etc/aide/aide.conf` or `/etc/aide/aide.conf.d/`.
   - Take a backup of the stock configuration files: `sudo cp -pr /etc/aide /etc/aide-COPY-$(date +"%Y%m%d%H%M%S")`.

6. Create a new database, and install it.

   On Debian based systems:

```
sudo aideinit
```

```
Running aide --init...
Start timestamp: 2019-04-01 21:23:37 -0400 (AIDE 0.16)
AIDE initialized database at /var/lib/aide/aide.db.new
Verbose level: 6

Number of entries:      25973

---------------------------------------------------
The attributes of the (uncompressed) database(s):
---------------------------------------------------

/var/lib/aide/aide.db.new
  RMD160   : moyQ1YskQQbidX+Lusv3g2wf1gQ=
  TIGER    : 7WoOgCrXzSpDrlO6I3PyXPj1gRiaMSeo
  SHA256   : gVx8Fp7r3800WF2aeXl+/KHCzfGsNi7O
             g16VTPpIfYQ=
  SHA512   : GYfa0DJwWgMLl4Goo5VFVOhu4BphXCo3
             rZnk49PYztwu50XjaAvsVuTjJY5uIYrG
             tV+jt3ELvwFzGefq4ZBNMg==
  CRC32    : /cusZw==
  HAVAL    : E/i5ceF3YTjwenBfyxHEsy9Kzu35VTf7
             CPGQSW4tl14=
  GOST     : n5Ityzxey9/1jIs7LMc08SULF1sLBFUc
             aMv7Oby604A=


End timestamp: 2019-04-01 21:24:45 -0400 (run time: 1m 8s)
```

7. Test everything works with no changes.

   On Debian based systems:

```
sudo aide.wrapper --check
```

```
Start timestamp: 2019-04-01 21:24:45 -0400 (AIDE 0.16)
AIDE found NO differences between database and filesystem. Looks okay!!
Verbose level: 6

Number of entries:      25973

---------------------------------------------------
The attributes of the (uncompressed) database(s):
---------------------------------------------------

/var/lib/aide/aide.db
  RMD160   : moyQ1YskQQbidX+Lusv3g2wf1gQ=
  TIGER    : 7WoOgCrXzSpDrlO6I3PyXPj1gRiaMSeo
  SHA256   : gVx8Fp7r3800WF2aeXl+/KHCzfGsNi7O
             g16VTPpIfYQ=
  SHA512   : GYfa0DJwWgMLl4Goo5VFVOhu4BphXCo3
```

```
              rZnk49PYztwu50XjaAvsVuTjJY5uIYrG
              tV+jt3ELvwFzGefq4ZBNMg==
  CRC32     : /cusZw==
  HAVAL     : E/i5ceF3YTjwenBfyxHEsy9Kzu35VTf7
              CPGQSW4tl14=
  GOST      : n5Ityzxey9/1jIs7LMc08SULF1sLBFUc
              aMv7Oby604A=


End timestamp: 2019-04-01 21:26:03 -0400 (run time: 1m 18s)
```

8. Test everything works after making some changes.

   On Debian based systems:

```
sudo touch /etc/test.sh
sudo touch /root/test.sh

sudo aide.wrapper --check

sudo rm /etc/test.sh
sudo rm /root/test.sh

sudo aideinit -y -f
```

```
Start timestamp: 2019-04-01 21:37:37 -0400 (AIDE 0.16)
AIDE found differences between database and filesystem!!
Verbose level: 6

Summary:
  Total number of entries:      25972
  Added entries:                2
  Removed entries:              0
  Changed entries:              1

---------------------------------------------------
Added entries:
---------------------------------------------------

f++++++++++++++++: /etc/test.sh
f++++++++++++++++: /root/test.sh

---------------------------------------------------
Changed entries:
---------------------------------------------------

d =.... mc.. .. .: /root

---------------------------------------------------
Detailed information about changes:
---------------------------------------------------

Directory: /root
  Mtime     : 2019-04-01 21:35:07 -0400        | 2019-04-01 21:37:36 -0400
  Ctime     : 2019-04-01 21:35:07 -0400        | 2019-04-01 21:37:36 -0400


---------------------------------------------------
The attributes of the (uncompressed) database(s):
---------------------------------------------------

/var/lib/aide/aide.db
  RMD160    : qF9WmKaf2PptjKnhcr9z4ueCPTY=
  TIGER     : zMo7MvvYJcq1hzvTQLPMW7ALeFiyEqv+
  SHA256    : LSLLVjjV6r8vlSxlbAbbEsPcQUB48SgP
              pdVqEn6ZNbQ=
  SHA512    : Qc4U7+ZAWCcitapGhJ1IrXCLGCf1IKZl
              02KYL1gaZ0Fm4dc7xLqjiquWDMSEbwzW
              oz49NCquqGz5jpMIUy7UxA==
  CRC32     : z8ChEA==
  HAVAL     : YapzS+/cdDwLj3kHJEq8fufLp3DPKZDg
              U12KCSkrO7Y=
  GOST      : 74sLV4HkTig+GJhokvxZQm7CJD/NR0mG
              6jV7zdt5AXQ=
```

```
End timestamp: 2019-04-01 21:38:50 -0400 (run time: 1m 13s)
```

9. That's it. If you set `CRON_DAILY_RUN` to `yes` in `/etc/default/aide` then cron will execute `/etc/cron.daily/aide` every day and e-mail you the output.

Updating The Database

Every time you make changes to files/folders that AIDE monitors, you will need to update the database to capture those changes. To do that on Debian based systems:

```
sudo aideinit -y -f
```

(Table of Contents)

Anti-Virus Scanning With ClamAV (WIP)

Why

WIP

How It Works

- ClamAV is a virus scanner
- ClamAV-Freshclam is a service that keeps the virus definitions updated
- ClamAV-Daemon keeps the `clamd` process running to make scanning faster

Goals

WIP

Notes

- These instructions **do not** tell you how to enable the ClamAV daemon service to ensure `clamd` is running all the time. `clamd` is only if you're running a mail server and does not provide real-time monitoring of files. Instead, you'd want to scan files manually or on a schedule.

References

- https://www.clamav.net/documents/installation-on-debian-and-ubuntu-linux-distributions
- https://wiki.debian.org/ClamAV
- https://www.osradar.com/install-clamav-debian-9-ubuntu-18/
- https://www.lisenet.com/2014/automate-clamav-to-perform-daily-system-scan-and-send-email-notifications-on-linux/
- https://www.howtoforge.com/tutorial/configure-clamav-to-scan-and-notify-virus-and-malware/
- https://serverfault.com/questions/741299/is-there-a-way-to-keep-clamav-updated-on-debian-8
- https://askubuntu.com/questions/250290/how-do-i-scan-for-viruses-with-clamav
- https://ngothang.com/how-to-install-clamav-and-configure-daily-scanning-on-centos/

Steps

1. Install ClamAV.

   On Debian based systems:

   ```
   sudo apt install clamav clamav-freshclam clamav-daemon
   ```

2. Make a backup of `clamav-freshclam` 's configuration file `/etc/clamav/freshclam.conf` :

   ```
   sudo cp --archive /etc/clamav/freshclam.conf /etc/clamav/freshclam.conf-COPY-$(date +"%Y%m%d%H%M%S")
   ```

3. `clamav-freshclam` 's default settings are probably good enough but if you want to change them, you can either edit the file `/etc/clamav/freshclam.conf` or use `dpkg-reconfigure` :

   ```
   sudo dpkg-reconfigure clamav-freshclam
   ```

Note: The default settings will update the definitions 24 times in a day. To change the interval, check the `Checks` setting in `/etc/clamav/freshclam.conf` or use `dpkg-reconfigure`.

4. Start the `clamav-freshclam` service:

```
sudo service clamav-freshclam start
```

5. You can make sure `clamav-freshclam` running:

```
sudo service clamav-freshclam status
```

```
● clamav-freshclam.service - ClamAV virus database updater
   Loaded: loaded (/lib/systemd/system/clamav-freshclam.service; enabled; vendor preset: enabled)   Active: active
(running) since Sat 2019-03-16 22:57:07 EDT; 2min 13s ago
     Docs: man:freshclam(1)
           man:freshclam.conf(5)
           https://www.clamav.net/documents
 Main PID: 1288 (freshclam)
   CGroup: /system.slice/clamav-freshclam.service
           └─1288 /usr/bin/freshclam -d --foreground=true

Mar 16 22:57:08 host freshclam[1288]: Sat Mar 16 22:57:08 2019 -> ^Local version: 0.100.2 Recommended version:
0.101.1
Mar 16 22:57:08 host freshclam[1288]: Sat Mar 16 22:57:08 2019 -> DON'T PANIC! Read
https://www.clamav.net/documents/upgrading-clamav
Mar 16 22:57:15 host freshclam[1288]: Sat Mar 16 22:57:15 2019 -> Downloading main.cvd [100%]
Mar 16 22:57:38 host freshclam[1288]: Sat Mar 16 22:57:38 2019 -> main.cvd updated (version: 58, sigs: 4566249, f-
level: 60, builder: sigmgr)
Mar 16 22:57:40 host freshclam[1288]: Sat Mar 16 22:57:40 2019 -> Downloading daily.cvd [100%]
Mar 16 22:58:13 host freshclam[1288]: Sat Mar 16 22:58:13 2019 -> daily.cvd updated (version: 25390, sigs: 1520006,
f-level: 63, builder: raynman)
Mar 16 22:58:14 host freshclam[1288]: Sat Mar 16 22:58:14 2019 -> Downloading bytecode.cvd [100%]
Mar 16 22:58:16 host freshclam[1288]: Sat Mar 16 22:58:16 2019 -> bytecode.cvd updated (version: 328, sigs: 94, f-
level: 63, builder: neo)
Mar 16 22:58:24 host freshclam[1288]: Sat Mar 16 22:58:24 2019 -> Database updated (6086349 signatures) from
db.local.clamav.net (IP: 104.16.219.84)
Mar 16 22:58:24 host freshclam[1288]: Sat Mar 16 22:58:24 2019 -> ^Clamd was NOT notified: Can't connect to clamd
through /var/run/clamav/clamd.ctl: No such file or directory
```

Note: Don't worry about that `Local version` line. Check https://serverfault.com/questions/741299/is-there-a-way-to-keep-clamav-updated-on-debian-8 for more details.

6. Make a backup of `clamav-daemon`'s configuration file `/etc/clamav/clamd.conf`:

```
sudo cp --archive /etc/clamav/clamd.conf /etc/clamav/clamd.conf-COPY-$(date +"%Y%m%d%H%M%S")
```

7. You can change `clamav-daemon`'s settings by editing the file `/etc/clamav/clamd.conf` or useing `dpkg-reconfigure`:

```
sudo dpkg-reconfigure clamav-daemon
```

Scanning Files/Folders

- To scan files/folders use the `clamscan` program.
- `clamscan` runs as the user it is executed as so it needs read permissions to the files/folders it is scanning.
- Using `clamscan` as `root` is dangerous because if a file is in fact a virus there is risk that it could use the root privileges.
- To scan a file: `clamscan /path/to/file`.
- To scan a directory: `clamscan -r /path/to/folder`.
- You can use the `-i` switch to only print infected files.
- Check `clamscan`'s `man` pages for other switches/options.

(Table of Contents)

Rootkit Detection With Rkhunter (WIP)

Why

WIP

How It Works

WIP

Goals

WIP

References

- http://rkhunter.sourceforge.net/
- https://www.cyberciti.biz/faq/howto-check-linux-rootkist-with-detectors-software/
- https://www.tecmint.com/install-rootkit-hunter-scan-for-rootkits-backdoors-in-linux/

Steps

1. Install Rkhunter.

   On Debian based systems:

   ```
   sudo apt install rkhunter
   ```

2. Make a backup of rkhunter' defaults file:

   ```
   sudo cp -p /etc/default/rkhunter /etc/default/rkhunter-COPY-$(date +"%Y%m%d%H%M%S")
   ```

3. rkhunter's configuration file is `/etc/rkhunter.conf`. Instead of making changes to it, create and use the file `/etc/rkhunter.conf.local` instead:

   ```
   sudo cp -p /etc/rkhunter.conf /etc/rkhunter.conf.local
   ```

4. Go through the configuration file `/etc/rkhunter.conf.local` and set to your requirements. My recommendations:

Setting	Note
`UPDATE_MIRRORS=1`	
`MIRRORS_MODE=0`	
`MAIL-ON-WARNING=root`	
`COPY_LOG_ON_ERROR=1`	to save a copy of the log if there is an error
`PKGMGR=...`	set to the appropriate value per the documentation
`PHALANX2_DIRTEST=1`	read the documentation for why
`WEB_CMD=""`	this is to address an issue with the Debian package that disables the ability for rkhunter to self-update.
`USE_LOCKING=1`	to prevent issues with rkhunter running multiple times
`SHOW_SUMMARY_WARNINGS_NUMBER=1`	to see the actual number of warnings found

5. You want rkhunter to run every day and e-mail you the result. You can write your own script or check https://www.tecmint.com/install-rootkit-hunter-scan-for-rootkits-backdoors-in-linux/ for a sample cron script you can use.

   On Debian based system, rkhunter comes with cron scripts. To enable them check `/etc/default/rkhunter` or use `dpkg-reconfigure` and say `Yes` to all of the questions:

   ```
   sudo dpkg-reconfigure rkhunter
   ```

6. After you've finished with all of the changes, make sure all the settings are valid:

   ```
   sudo rkhunter -C
   ```

7. Update rkhunter and its database:

```
sudo rkhunter --versioncheck
sudo rkhunter --update
sudo rkhunter --propupd
```

8. If you want to do a manual scan and see the output:

```
sudo rkhunter --check
```

(Table of Contents)

Rootkit Detection With chrootkit (WIP)

Why

WIP

How It Works

WIP

Goals

WIP

References

- http://www.chkrootkit.org/
- https://www.cyberciti.biz/faq/howto-check-linux-rootkist-with-detectors-software/
- https://askubuntu.com/questions/258658/eth0-packet-sniffer-sbin-dhclient

Steps

1. Install chkrootkit.

   On Debian based systems:

   ```
   sudo apt install chkrootkit
   ```

2. Do a manual scan:

   ```
   sudo chkrootkit
   ```

   ```
   ROOTDIR is `/'
   Checking `amd'...                                   not found
   Checking `basename'...                              not infected
   Checking `biff'...                                  not found
   Checking `chfn'...                                  not infected
   Checking `chsh'...                                  not infected
   ...
   Checking `scalper'...                               not infected
   Checking `slapper'...                               not infected
   Checking `z2'...                                    chklastlog: nothing deleted
   Checking `chkutmp'...                               chkutmp: nothing deleted
   Checking `OSX_RSPLUG'...                            not infected
   ```

3. Make a backup of chkrootkit's configuration file `/etc/chkrootkit.conf`:

   ```
   sudo cp --archive /etc/chkrootkit.conf /etc/chkrootkit.conf-COPY-$(date +"%Y%m%d%H%M%S")
   ```

4. You want chkrootkit to run every day and e-mail you the result.

   On Debian based system, chkrootkit comes with cron scripts. To enable them check `/etc/chkrootkit.conf` or use `dpkg-reconfigure` and say `Yes` to the first question:

   ```
   sudo dpkg-reconfigure chkrootkit
   ```

(Table of Contents)

logwatch - system log analyzer and reporter

Why

Your server will be generating a lot of logs that may contain important information. Unless you plan on checking your server everyday, you'll want a way to get e-mail summary of your server's logs. To accomplish this we'll use logwatch.

How It Works

logwatch scans system log files and summarizes them. You can run it directly from the command line or schedule it to run on a recurring schedule. logwatch uses service files to know how to read/summarize a log file. You can see all of the stock service files in `/usr/share/logwatch/scripts/services`.

logwatch's configuration file `/usr/share/logwatch/default.conf/logwatch.conf` specifies default options. You can override them via command line arguments.

Goals

- Logwatch configured to send a daily e-mail summary of all of the server's status and logs

Notes

- Your server will need to be able to send e-mails for this to work
- The below steps will result in logwatch running every day. If you want to change the schedule, modify the cronjob to your liking. You'll also want to change the `range` option to cover your recurrence window. See https://www.badpenguin.org/configure-logwatch-for-weekly-email-and-html-output-format for an example.
- If logwatch fails to deliver mail due to the e-mail having long lines please check https://blog.dhampir.no/content/exim4-line-length-in-debian-stretch-mail-delivery-failed-returning-message-to-sender as documented in issue #29. If you followed Gmail and Exim4 As MTA With Implicit TLS then we already took care of this in step #7.

References

- Thanks to amacheema for fixing some issues with the steps and letting me know of a long line bug with exim4 as documented in issue #29.
- https://sourceforge.net/projects/logwatch/
- https://www.digitalocean.com/community/tutorials/how-to-install-and-use-logwatch-log-analyzer-and-reporter-on-a-vps

Steps

1. Install logwatch.

   On Debian based systems:

   ```
   sudo apt install logwatch
   ```

2. To see a sample of what logwatch collects you can run it directly:

   ```
   sudo /usr/sbin/logwatch --output stdout --format text --range yesterday --service all
   ```

   ```
   ################### Logwatch 7.4.3 (12/07/16) ####################
          Processing Initiated: Mon Mar  4 00:05:50 2019
          Date Range Processed: yesterday
                                ( 2019-Mar-03 )
                                Period is day.
          Detail Level of Output: 5
          Type of Output/Format: stdout / text
          Logfiles for Host: host
   ##################################################################

    --------------------- Cron Begin ------------------------
   ...
   ...
    ---------------------- Disk Space End -------------------------


    ###################### Logwatch End #########################
   ```

3. Go through logwatch's self-documented configuration file `/usr/share/logwatch/default.conf/logwatch.conf` before continuing. There is no need to change anything here but pay special attention to the `Output`, `Format`, `MailTo`, `Range`, and `Service` as those are the ones we'll be using. For our purposes, instead of specifying our options in the configuration file, we will pass them as command line arguments in the daily cron job that executes logwatch. That way, if the configuration file is ever modified (e.g. during an update), our options will still be there.

4. Make a backup of logwatch's daily cron file `/etc/cron.daily/00logwatch` and unset the execute bit:

```
sudo cp --archive /etc/cron.daily/00logwatch /etc/cron.daily/00logwatch-COPY-$(date +"%Y%m%d%H%M%S")
sudo chmod -x /etc/cron.daily/00logwatch-COPY*
```

5. By default, logwatch outputs to `stdout`. Since the goal is to get a daily e-mail, we need to change the output type that logwatch uses to send e-mail instead. We could do this through the configuration file above, but that would apply to every time it is run -- even when we run it manually and want to see the output to the screen. Instead, we'll change the cron job that executes logwatch to send e-mail. This way, when run manually, we'll still get output to `stdout` and when run by cron, it'll send an e-mail. We'll also make sure it checks for all services, and change the output format to html so it's easier to read regardless of what the configuration file says. In the file `/etc/cron.daily/00logwatch` find the execute line and change it to:

```
/usr/sbin/logwatch --output mail --format html --mailto root --range yesterday --service all
```

```
#!/bin/bash

#Check if removed-but-not-purged
test -x /usr/share/logwatch/scripts/logwatch.pl || exit 0

#execute
/usr/sbin/logwatch --output mail --format html --mailto root --range yesterday --service all

#Note: It's possible to force the recipient in above command
#Just pass --mailto address@a.com instead of --output mail
```

For the lazy:

```
sudo sed -i -r -e "s,^($(sudo which logwatch).*?),# \1         # commented by $(whoami) on $(date +"%Y-%m-%d @ %H:%M:%S")\n
```

6. You can test the cron job by executing it:

```
sudo /etc/cron.daily/00logwatch
```

Note: If logwatch fails to deliver mail due to the e-mail having long lines please check https://blog.dhampir.no/content/exim4-line-length-in-debian-stretch-mail-delivery-failed-returning-message-to-sender as documented in issue #29. If you followed Gmail and Exim4 As MTA With Implicit TLS then we already took care of this in step #7.

(Table of Contents)

ss - Seeing Ports Your Server Is Listening On

Why

Ports are how applications, services, and processes communicate with each other -- either locally within your server or with other devices on the network. When you have an application or service (like SSH or Apache) running on your server, they listen for requests on specific ports.

Obviously we don't want your server listening on ports we don't know about. We'll use `ss` to see all the ports that services are listening on. This will help us track down and stop rogue, potentially dangerous, services.

Goals

- find out non-localhost what ports are open and listening for connections

References

- https://www.reddit.com/r/linux/comments/arx7st/howtosecurealinuxserver_an_evolving_howto_guide/egrib6o/
- https://www.reddit.com/r/linux/comments/arx7st/howtosecurealinuxserver_an_evolving_howto_guide/egs1rev/
- https://www.tecmint.com/find-open-ports-in-linux/
- `man ss`

Steps

1. To see the all the ports listening for traffic:

```
sudo ss -lntup
```

```
Netid  State      Recv-Q Send-Q     Local Address:Port     Peer Address:Port
udp    UNCONN     0      0                     *:68                  *:*        users:(("dhclient",pid=389,fd=6))
tcp    LISTEN     0      128                   *:22                  *:*        users:(("sshd",pid=4390,fd=3))
tcp    LISTEN     0      128                  :::22                 :::*        users:(("sshd",pid=4390,fd=4))
```

Switch Explanations:

- `l` = display listening sockets
- `n` = do not try to resolve service names
- `t` = display TCP sockets
- `u` = display UDP sockets
- `p` = show process information

2. If you see anything suspicious, like a port you're not aware of or a process you don't know, investigate and remediate as necessary.

(Table of Contents)

Lynis - Linux Security Auditing

Why

From https://cisofy.com/lynis/:

> Lynis is a battle-tested security tool for systems running Linux, macOS, or Unix-based operating system. It performs an extensive health scan of your systems to support system hardening and compliance testing.

Goals

- Lynis installed

Notes

- CISOFY offers packages for many distributions. Check https://packages.cisofy.com/ for distribution specific installation instructions.

References

- https://cisofy.com/documentation/lynis/get-started/
- https://packages.cisofy.com/community/#debian-ubuntu
- https://thelinuxcode.com/audit-lynis-ubuntu-server/
- https://www.vultr.com/docs/install-lynis-on-debian-8

Steps

1. Install lynis. https://cisofy.com/lynis/#installation has detailed instructions on how to install it for your distribution.

   On Debian based systems, using CISOFY's community software repository:

```
sudo apt install apt-transport-https ca-certificates host
sudo wget -O - https://packages.cisofy.com/keys/cisofy-software-public.key | sudo apt-key add -
sudo echo "deb https://packages.cisofy.com/community/lynis/deb/ stable main" | sudo tee /etc/apt/sources.list.d/cisofy-lyni
sudo apt update
sudo apt install lynis host
```

2. Update it:

```
sudo lynis update info
```

3. Run a security audit:

```
sudo lynis audit system
```

This will scan your server, report its audit findings, and at the end it will give you suggestions. Spend some time going through the output and address gaps as necessary.

(Table of Contents)

OSSEC - Host Intrusion Detection

Why

From https://github.com/ossec/ossec-hids

> OSSEC is a full platform to monitor and control your systems. It mixes together all the aspects of HIDS (host-based intrusion detection), log monitoring and SIM/SIEM together in a simple, powerful and open source solution.

Goals

- OSSEC-HIDS installed

References

- https://www.ossec.net/docs/

Steps

1. Install OSSEC-HIDS from sources

```
sudo apt install -y libz-dev libssl-dev libpcre2-dev build-essential libsystemd-dev
wget https://github.com/ossec/ossec-hids/archive/3.7.0.tar.gz
tar xzf 3.7.0.tar.gz
cd ossec-hids-3.7.0/
sudo ./install.sh
```

2. Useful commands:

Agent information

```
sudo /var/ossec/bin/agent_control -i <AGENT_ID>
```

`AGENT_ID` by default is `000`, to be sure the command `sudo /var/ossec/bin/agent_control -l` can be used.

Run integrity/rootkit checking

OSSEC by default run rootkit check each 2 hours.

```
sudo /var/ossec/bin/agent_control -u <AGENT_ID> -r
```

Alerts

- All:

```
tail -f /var/ossec/logs/alerts/alerts.log
```

- Integrity check:

```
sudo cat /var/ossec/logs/alerts/alerts.log | grep -A4  -i integrity
```

- Rootkit check:

```
sudo cat /var/ossec/logs/alerts/alerts.log | grep -A4  "rootcheck,"
```

(Table of Contents)

The Danger Zone

Proceed At Your Own Risk

This sections cover things that are high risk because there is a possibility they can make your system unusable, or are considered unnecessary by many because the risks outweigh any rewards.

!! PROCEED AT YOUR OWN RISK !!

▶ !! PROCEED AT YOUR OWN RISK !!

(Table of Contents)

The Miscellaneous

The Simple way with MSMTP

(#msmtp-alternative)

Why

Well I will SIMPLIFY this method, to only output email using Google Mail account (and others). True Simple! :)

```
``` bash
#!/bin/bash
###### PLEASE .... EDIT IT...
USEREMAIL="usernameemail"
DOMPROV="gmail.com"
PWDEMAIL="passwordStrong"  ## ATTENTION DONT USE Special Chars.. like as SPACE # and some others not all. Feel free to test
;)
MAILPROV="smtp.google.com:583"
MYMAIL="$USRMAIL@$DOMPROV"
USERLOC="root"
#######
apt install -y msmtp
    ln -s /usr/bin/msmtp /usr/sbin/sendmail
#wget http://www.cacert.org/revoke.crl -O /etc/ssl/certs/revoke.crl
#chmod 644 /etc/ssl/certs/revoke.crl
touch /root/.msmtprc
cat <<EOF> .msmtprc
defaults
account gmail
host $MAILPROV
port $MAILPORT
#proxy_host 127.0.0.1
#proxy_port 9001
from $MYEMAIL
timeout off
protocol smtp
#auto_from [(on|off)]
#from envelope_from
#maildomain [domain]
auth on
user $USRMAIL
passwordeval "gpg -q --for-your-eyes-only --no-tty -d /root/msmtp-mail.gpg"
#passwordeval "gpg --quiet --for-your-eyes-only --no-tty --decrypt /root/msmtp-mail.gpg"
tls on
tls_starttls on
tls_trust_file /etc/ssl/certs/ca-certificates.crt
#tls_crl_file /etc/ssl/certs/revoke.crl
#tls_fingerprint [fingerprint]
#tls_key_file [file]
#tls_cert_file [file]
tls_certcheck on
tls_force_sslv3 on
tls_min_dh_prime_bits 512
#tls_priorities [priorities]
#dsn_notify (off|condition)
#dsn_return (off|amount)
#domain argument
#keepbcc off
logfile /var/log/mail.log
syslog on
account default : gmail
EOF
chmod 0400 /root/.msmtprc

   ## In testing .. auto command
# echo -e "1\n4096\n\ny\n$MYUSRMAIL\n$MYEMAIL\nmy key\nO\n$PWDMAIL\n$PWDMAIL\n" | gpg --full-generate-key
##
gpg --full-generate-key
gpg --output revoke.asc --gen-revoke $MYEMAIL
```
```

```
echo -e "$PWDEMAIL\n" | gpg -e -o /root/msmtp-mail.gpg --recipient $MYEMAIL
echo "export GPG_TTY=\$(tty)" >> .baschrc
chmod 400 msmtp-mail.gpg

echo "Hello there" | msmtp --debug $MYEMAIL
echo"######################
## MSMTP Configured ##
######################"
```

DONE!! ;) (Table of Contents)

Gmail and Exim4 As MTA With Implicit TLS

Why

Unless you're planning on setting up your own mail server, you'll need a way to send e-mails from your server. This will be important for system alerts/messages.

You can use any Gmail account. I recommend you create one specific for this server. That way if your server **is** compromised, the bad-actor won't have any passwords for your primary account. Granted, if you have 2FA/MFA enabled, and you use an app password, there isn't much a bad-actor can do with just the app password, but why take the risk?

There are many guides on-line that cover how to configure Gmail as MTA using STARTTLS including a previous version of this guide. With STARTTLS, an initial **unencrypted** connection is made and then upgraded to an encrypted TLS or SSL connection. Instead, with the approach outlined below, an encrypted TLS connection is made from the start.

Also, as discussed in issue #29 and here, exim4 will fail for messages with long lines. We'll fix this in this section too.

** **IMPORTANT** ** As mentioned in #106, Google no longer lets you use your account's password for authentication. You have to enable 2FA and then use an app-password.

Goals

- `mail` configured to send e-mails from your server using Gmail
- long line support for exim4

References

- Thanks to remyabel for figuring out how to get this to work with TLS as documented in issue #24 and pull request #26.
- https://wiki.debian.org/Exim
- https://wiki.debian.org/GmailAndExim4
- https://www.exim.org/exim-html-current/doc/html/spec_html/ch-encrypted_smtp_connections_using_tlssl.html
- https://php.quicoto.com/setup-exim4-to-use-gmail-in-ubuntu/
- https://www.fastmail.com/help/technical/ssltlsstarttls.html
- exim4 fails for messages with long lines - issue #29 and https://blog.dhampir.no/content/exim4-line-length-in-debian-stretch-mail-delivery-failed-returning-message-to-sender
- #106

Steps

1. Install exim4. You will also need openssl and ca-certificates.

   On Debian based systems:

   ```
   sudo apt install exim4 openssl ca-certificates
   ```

2. Configure exim4:

   For Debian based systems:

   ```
   sudo dpkg-reconfigure exim4-config
   ```

   You'll be prompted with some questions:

Prompt	Answer
General type of mail configuration	`mail sent by smarthost; no local mail`

Prompt	Answer
System mail name	`localhost`
IP-addresses to listen on for incoming SMTP connections	`127.0.0.1; ::1`
Other destinations for which mail is accepted	(default)
Visible domain name for local users	`localhost`
IP address or host name of the outgoing smarthost	`smtp.gmail.com::465`
Keep number of DNS-queries minimal (Dial-on-Demand)?	`No`
Split configuration into small files?	`No`

3. Make a backup of `/etc/exim4/passwd.client` :

```
sudo cp --archive /etc/exim4/passwd.client /etc/exim4/passwd.client-COPY-$(date +"%Y%m%d%H%M%S")
```

4. Add a line like this to `/etc/exim4/passwd.client`

```
smtp.gmail.com:yourAccount@gmail.com:yourPassword
*.google.com:yourAccount@gmail.com:yourPassword
```

Notes:

- Replace `yourAccount@gmail.com` and `yourPassword` with your details. If you have 2FA/MFA enabled on your Gmail then you'll need to create and use an app password here.
- Always check `host smtp.gmail.com` for the most up-to-date domains to list.

5. This file has your Gmail password so we need to lock it down:

```
sudo chown root:Debian-exim /etc/exim4/passwd.client
sudo chmod 640 /etc/exim4/passwd.client
```

6. The next step is to create an TLS certificate that exim4 will use to make the encrypted connection to `smtp.gmail.com` . You can use your own certificate, like one from Let's Encrypt, or create one yourself using openssl. We will use a script that comes with exim4 that calls openssl to make our certificate:

```
sudo bash /usr/share/doc/exim4-base/examples/exim-gencert
```

```
[*] Creating a self signed SSL certificate for Exim!
    This may be sufficient to establish encrypted connections but for
    secure identification you need to buy a real certificate!

    Please enter the hostname of your MTA at the Common Name (CN) prompt!

Generating a RSA private key
.......................................+++++
...............................................+++++
writing new private key to '/etc/exim4/exim.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Code (2 letters) [US]:[redacted]
State or Province Name (full name) []:[redacted]
Locality Name (eg, city) []:[redacted]
Organization Name (eg, company; recommended) []:[redacted]
Organizational Unit Name (eg, section) []:[redacted]
Server name (eg. ssl.domain.tld; required!!!) []:localhost
Email Address []:[redacted]
[*] Done generating self signed certificates for exim!
    Refer to the documentation and example configuration files
    over at /usr/share/doc/exim4-base/ for an idea on how to enable TLS
    support in your mail transfer agent.
```

7. Instruct exim4 to use TLS and port 465, and fix exim4's long lines issue, by creating the file `/etc/exim4/exim4.conf.localmacros` and adding:

```
MAIN_TLS_ENABLE = 1
REMOTE_SMTP_SMARTHOST_HOSTS_REQUIRE_TLS = *
TLS_ON_CONNECT_PORTS = 465
REQUIRE_PROTOCOL = smtps
IGNORE_SMTP_LINE_LENGTH_LIMIT = true
```

For the lazy:

```
cat << EOF | sudo tee /etc/exim4/exim4.conf.localmacros
MAIN_TLS_ENABLE = 1
REMOTE_SMTP_SMARTHOST_HOSTS_REQUIRE_TLS = *
TLS_ON_CONNECT_PORTS = 465
REQUIRE_PROTOCOL = smtps
IGNORE_SMTP_LINE_LENGTH_LIMIT = true
EOF
```

8. Make a backup of exim4's configuration file `/etc/exim4/exim4.conf.template`:

```
sudo cp --archive /etc/exim4/exim4.conf.template /etc/exim4/exim4.conf.template-COPY-$(date +"%Y%m%d%H%M%S")
```

9. Add the below to `/etc/exim4/exim4.conf.template` after the `.ifdef REMOTE_SMTP_SMARTHOST_HOSTS_REQUIRE_TLS ... .endif` block:

```
.ifdef REQUIRE_PROTOCOL
  protocol = REQUIRE_PROTOCOL
.endif
```

> ```
> .ifdef REMOTE_SMTP_SMARTHOST_HOSTS_REQUIRE_TLS
>   hosts_require_tls = REMOTE_SMTP_SMARTHOST_HOSTS_REQUIRE_TLS
> .endif
> .ifdef REQUIRE_PROTOCOL
>     protocol = REQUIRE_PROTOCOL
> .endif
> .ifdef REMOTE_SMTP_HEADERS_REWRITE
>   headers_rewrite = REMOTE_SMTP_HEADERS_REWRITE
> .endif
> ```

For the lazy:

```
sudo sed -i -r -e '/^.ifdef REMOTE_SMTP_SMARTHOST_HOSTS_REQUIRE_TLS$/I { :a; n; /^.endif$/!ba; a\# added by '"$(whoami) on
```

10. Add the below to `/etc/exim4/exim4.conf.template` inside the `.ifdef MAIN_TLS_ENABLE` block:

```
.ifdef TLS_ON_CONNECT_PORTS
  tls_on_connect_ports = TLS_ON_CONNECT_PORTS
.endif
```

> ```
> .ifdef MAIN_TLS_ENABLE
> .ifdef TLS_ON_CONNECT_PORTS
>     tls_on_connect_ports = TLS_ON_CONNECT_PORTS
> .endif
> ```

For the lazy:

```
sudo sed -i -r -e "/\.ifdef MAIN_TLS_ENABLE/ a # added by $(whoami) on $(date +"%Y-%m-%d @ %H:%M:%S")\n.ifdef TLS_ON_CONNEC
```

11. Update exim4 configuration to use TLS and then restart the service:

```
sudo update-exim4.conf
sudo service exim4 restart
```

12. If you're using UFW, you'll need to allow outbound traffic on 465. To do this we'll create a custom UFW application profile and then enable it. Create the file `/etc/ufw/applications.d/smtptls`, add this, then run `ufw allow out smtptls comment 'open TLS port 465 for use with SMPT to send e-mails'`:

    ```
    [SMTPTLS]
    title=SMTP through TLS
    description=This opens up the TLS port 465 for use with SMPT to send e-mails.
    ports=465/tcp
    ```

    For the lazy:

    ```
    cat << EOF | sudo tee /etc/ufw/applications.d/smtptls
    [SMTPTLS]
    title=SMTP through TLS
    description=This opens up the TLS port 465 for use with SMPT to send e-mails.
    ports=465/tcp
    EOF

    sudo ufw allow out smtptls comment 'open TLS port 465 for use with SMPT to send e-mails'
    ```

13. Add some mail aliases so we can send e-mails to local accounts by adding lines like this to `/etc/aliases`:

    ```
    user1: user1@gmail.com
    user2: user2@gmail.com
    ...
    ```

    You'll need to add all the local accounts that exist on your server.

14. Test your setup:

    ```
    echo "test" | mail -s "Test" email@gmail.com
    sudo tail /var/log/exim4/mainlog
    ```

(Table of Contents)

Separate iptables Log File

Why

There will come a time when you'll need to look through your iptables logs. Having all the iptables logs go to their own file will make it a lot easier to find what you're looking for.

References

- https://blog.shadypixel.com/log-iptables-messages-to-a-separate-file-with-rsyslog/
- https://gist.github.com/netson/c45b2dc4e835761fbccc
- https://www.rsyslog.com/doc/v8-stable/configuration/actions.html

Steps

1. The first step is by telling your firewall to prefix all log entries with some unique string. If you're using iptables directly, you would do something like `--log-prefix "[IPTABLES] "` for all the rules. We took care of this in step 4 of installing psad.

2. After you've added a prefix to the firewall logs, we need to tell rsyslog to send those lines to its own file. Do this by creating the file `/etc/rsyslog.d/10-iptables.conf` and adding this:

    ```
    :msg, contains, "[IPTABLES] " /var/log/iptables.log
    & stop
    ```

    If you're expecting a lot if data being logged by your firewall, prefix the filename with a `-` "to omit syncing the file after every logging". For example:

    ```
    :msg, contains, "[IPTABLES] " -/var/log/iptables.log
    & stop
    ```

    Note: Remember to change the prefix to whatever you use.

For the lazy:

```
cat << EOF | sudo tee /etc/rsyslog.d/10-iptables.conf
:msg, contains, "[IPTABLES] " /var/log/iptables.log
& stop
EOF
```

3. Since we're logging firewall messages to a different file, we need to tell psad where the new file is. Edit `/etc/psad/psad.conf` and set `IPT_SYSLOG_FILE` to the path of the log file. For example:

```
IPT_SYSLOG_FILE /var/log/iptables.log;
```

Note: Remember to change the prefix to whatever you use.

For the lazy:

```
sudo sed -i -r -e "s/^(IPT_SYSLOG_FILE\s+)([^;]+)(;)$/# \1\2\3       # commented by $(whoami) on $(date +"%Y-%m-%d @ %H:%M:
```

4. Restart psad and rsyslog to activate the changes (or reboot):

```
sudo psad -R
sudo psad --sig-update
sudo psad -H
sudo service rsyslog restart
```

5. The last thing we have to do is tell logrotate to rotate the new log file so it doesn't get to big and fill up our disk. Create the file `/etc/logrotate.d/iptables` and add this:

```
/var/log/iptables.log
{
    rotate 7
    daily
    missingok
    notifempty
    delaycompress
    compress
    postrotate
        invoke-rc.d rsyslog rotate > /dev/null
    endscript
}
```

For the lazy:

```
cat << EOF | sudo tee /etc/logrotate.d/iptables
/var/log/iptables.log
{
    rotate 7
    daily
    missingok
    notifempty
    delaycompress
    compress
    postrotate
        invoke-rc.d rsyslog rotate > /dev/null
    endscript
}
EOF
```

(Table of Contents)

Left Over

Contacting Me

For any questions, comments, concerns, feedback, or issues, submit a new issue.

(Table of Contents)

Helpful Links

- https://github.com/pratiktri/server_init_harden - Bash script that automates few of the tasks that you need to perform on a new Linux server to give it basic amount security.

(Table of Contents)

Acknowledgments

- https://www.reddit.com/r/linuxquestions/comments/aopzl7/new_guide_created_by_me_how_to_secure_a_linux/
- https://www.reddit.com/r/selfhosted/comments/aoxd4l/new_guide_created_by_me_how_to_secure_a_linux/
- https://news.ycombinator.com/item?id=19177435#19178618

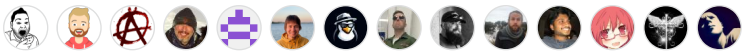
Releases

No releases published

Packages

No packages published

Contributors 29



+ 15 contributors