Autobahn security

Cyber Fitness Workout

Secure SSH

Secure SSH

Objective

Today we are hardening SSH. There are 7 steps ahead of you to properly secure your SSH:

1. Migrate to key-based authentication
2. Disable password-based authentication
3. Disable weak key exchange algorithms, ciphers, and message authentication codes (MACs)
4. Disable SSHv1
5. Disable the root login
6. Set a custom SSH port
7. Restrict SSH access using iptables

The first 4 steps remediate the finding, the other 3 steps further improve your SSH security.

About

Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. Using SSH with public keys is generally considered more secure than passwords.

Risk

When using password-based SSH, adversaries can brute-force your login credentials and gain access to the node. Besides that, exposing your SSH service broadens your attack surface.

Older versions of SSH, weak ciphers, key exchange algorithms, or message authentication codes can allow an attacker to recover plain text messages.

Fitness Workout

Throughout this workout we have used vim as editor. Feel free to use any editor you prefer.

→ START REMEDIATION

01. STEP
Migrate to key-based authentication

To migrate to key-based authentication, generate a key pair (a private and public key) on your computer. It is important that each computer that is — and that will be — using SSH creates a personal key pair. This means that this should be done per workstation.

✓ 1.1 - Check if you already have a key pair on your workstation

To check, run the following command in the terminal:

```
ls -al ~/.ssh/id_*.pub
```

If you already have a key pair, you can use it for authentication. Otherwise, proceed to create new keys as explained in the next step.

✓ 1.2 - Generate a new key pair

Run the following command in the terminal:

```
ssh-keygen -t rsa
```

You will get a notification **"generating public/private RSA key pair"**. Once done, click **Enter** and accept the default location.

You can change the location by typing in a new path. However, it is recommended that you make use of the default location. This is to avoid making unnecessary changes to your SSH client

You will be given the option to insert a **passphrase**, which serves as an extra layer of security. This is optional, so if you do not want to do this, click **Enter** to continue. After clicking **Enter**, the keys will be created

✓ 1.3 - List your keys

After creating the keys, list them by running the following command in the terminal:

```
ls ~/.ssh/id_*
```

✓ 1.4 - Copy the keys to the remote server

Next, copy the keys to the server you want to access. To do this, run the following command in the terminal:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub remote_username@server_ip_address
```

Change the **server_ip_address** to the server's IP address, and change **remote_username** to the correct username. The remote_username's password will be requested. When you type in the correct password, the public key will be saved on the server.

✓ 1.5 - Log in to your remote server

Now log in to the server without a password by running the following command in the terminal:

```
ssh remote_username@server_ip_address
```

02. STEP
Disable password-based authentication

If a workstation's private key is lost and password-based authentication is disabled, you have locked yourself out of the sever. It might be smart to skip this step if you don't have physical access to server interface

If you already use public key based authentication, go to step 2.3 and remove password authentication for all your servers because it is not necessary any more. If you do not yet use it, you need to go through all the steps below.

✓ 2.1 - Log in to the server using SSH keys

```
ssh remote_username@server_ip_address
```

✓ 2.2 - Open the SSH configuration either as a root user or as a user with sudo privileges

Run the following command in the terminal:

```
sudo vim /etc/ssh/sshd_config
```

✓ 2.3 - Edit the PasswordAuthentication line

Edit the file with your preferred text editor and set the `PasswordAuthentication` to `no` as shown below:

```
PasswordAuthentication no
```

Do not forget to uncomment the line if the # is present

✓ 2.4 - Restart the SSH service to apply the new configuration

```
sudo systemctl restart sshd
```

03. STEP Disable weak Key Exchanges, Ciphers, and MACs

To preserve backwards compatibility, your SSH server might use weak Key Exchange algorithms, ciphers, or message authentication codes. You should specifically set the KEXs, ciphers, and MACs that your server allows.

✓ 3.1 - Log in to the server using SSH keys

```
ssh remote_username@server_ip_address
```

✓ 3.2 - Open the SSH configuration either as a root user or as a user with sudo privileges

```
sudo vim /etc/ssh/sshd_config
```

✓ 3.3 - Set the values for "KexAlgorithms", "MACs", "Ciphers"

Edit the file with your preferred text editor and set the `KexAlgorithms` , `MACs` ,and `Ciphers` as shown below:

```
KexAlgorithms curve25519-sha256@libssh.org,ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-
nistp256,diffie-hellman-group-exchange-sha2 MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-
etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-512,hmac-sha2-256,umac-128@openssh Ciphers
chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-gcm@openssh.com,aes256-ctr,aes192-
ctr,aes128-ctr
```

✓ 3.4 - Restart the SSH service to apply the new configuration

```
sudo systemctl restart sshd
```

04. STEP Disable the usage of SSHv1

To preserve backwards compatibility, your SSH server might use SSHv1. Disable it for enhanced security.

✓ 4.1 - Log in to the server using SSH keys

```
ssh remote_username@server_ip_address
```

✓ 4.2 - Open the SSH configuration either as a root user or as a user with sudo privileges

```
sudo vim /etc/ssh/sshd_config
```

✓ 4.3 - Specify just `Protocol 2`

Find the line in `sshd_config` file where `Protocol` word is specified. Examples could be:

`Protocol 2,1`
or
`#Protocol 2,1`

Replace the line with `Protocol 2`

✓ 4.4 - Restart the SSH service to apply the new configuration

```
sudo systemctl restart sshd
```

05. STEP / OPTIONAL Disable the root login

Enabling direct login to root via SSH is a serious security threat as adversaries can brute-force your password. Before remediation ensure administrator rights has beed set up properly.

✓ 5.1 - Assign sudo rights

As root or sudo user type:

```
sudo visudo
```

Append the following line to end of the file:

```
<user_name> ALL=(ALL) NOPASSWD:ALL
```

The code snippet above enables users to execute code as root user without a password. Groups can be added if needed. Remove `NOPASSWD:ALL` if the password should be typed every time a user uses the sudo command.

✓ 5.2 - Log in as root and open the main configuration file

As root, edit the `sshd_config` file in `/etc/ssh/sshd_config` by running the following command in the terminal:

```
sudo vim /etc/ssh/sshd_config
```

✓ 5.3 - Search for the "PermitRootLogin yes" and replace the `yes` with `no`

Edit the file with your preferred editor as shown below:

```
PermitRootLogin no
```

✓ 5.4 - Restart the SSH service to apply the new configuration

```
sudo systemctl restart sshd
```

06. STEP / OPTIONAL
Set a custom SSH port

Port 22 is the default port for SSH connections. As is well-known, the majority of adversaries target this port. Changing the default SSH port helps mitigate automated attacks and wards off attackers from trying to access your SSH server.

✓ 6.1 - Choose the port you will use

It is recommended to use a random port between **49152** and **65535**. This is the range for the so-called "dynamic ports" (also called private ports)

✓ 6.2 - Login to the server and run the following command in the terminal

```
sudo vim /etc/ssh/sshd_config
```

✓ 6.3 - Change the default port to a preferred custom port

First, find the `Port 22` line. Then use your preferred editor to change this default port 22 to your preferred custom port.

✓ 6.4 - Restart your SSH service to apply the changes

```
sudo systemctl restart sshd
```

SSH is now listening on the custom port you specified.

Your client only knows the default SSH port. Consequently, you have to specify your custom port when connecting

07. STEP / OPTIONAL
Restrict SSH access using iptables

Next, we filter incoming and outgoing traffic to your server. You can set rules that restrict or permit traffic based on IP addresses and other related parameters. This will help to keep away unwanted traffic and minimize brute-force attacks.

For the sake of demonstration, we will set rules to permit incoming traffic for both one IP address and a range of IP addresses. You can replace the example IP address(es) with your own IP address(es).

✓ 7.1 - Allowlist an IP address of your choice

SSH into your server and execute the following command:

Allow one IP address:

```
sudo iptables -A INPUT -p tcp -s 192.1.1.1 -dport 22 -j ACCEPT
```

Allow a range of IP addresses:

```
iptables -A INPUT -i eth1 -m iprange --src-range 192.1.1.10-192.1.1.20 -dport 22 -j ACCEPT
```

Don not forget to change the example IP `sshd_config`, interface `eth1`, and port `22` to the IP that you want to allowlist and the port that your SSH server is lestening to

7.2 - Save the previously defined rule

Run the following command in the terminal:

```
sudo iptables-save
```

The commands are case sensitive. In addition, if for any reason you want to reset or get rid of all rules, use the command: **iptables -F**

Great job, you're in control now! Thank you for doing this important workout with us.

Get cyber-fit in just 3 months!