Casper Validator Node Security Framework

Last Updated: July 6, 2021

Introduction

The purpose of this document is to provide instruction and in some cases guidance regarding the security of a Casper Node. This document will outline any changes that are made from the standard basic configurations that are given Official CasperLabs Operators Documentation which can be found here (link to Official Documentation).

Open Ports

There are a number of ports that must remain exposed to the public. These ports are used and referenced in this document to inspect the node and do various tasks.

- 7777 - RPC Port
- 8888 - Status Port
- 9999 - Event Port
- 35000 - Gossip Port
- 3987 - ssh port used in this document

Security Features & Checklist

Shown on the next page is a basic checklist of all the security features that should be implemented on the Casper Validator Node. This checklist can be printed and filled out by a SysAdmin or a DevOps team. The security features themselves are described and implemented in the following sections.

Security Checklist

If any portion of this checklist cannot be completed, the SysAdmin or DevOps team should comment on the reasons why and write what steps were taken to mitigate the security risk.

If no steps were taken to mitigate the security risk, the operator should comment "risk understood & accepted" and the feature should be left unchecked since it has not been implemented.

1. Login and root user
    - ❑ Change the ssh port
    - ❑ Disable ssh root user login
    - ❑ Disable ssh password authentication
    - ❑ Disable root account
    - ❑ Non-root user login with ssh key
    - ❑ MFA - Specifically 2FA (Two - Factor Authentication)

2. Firewall
    - ❑ Block all unused ports
    - ❑ Verify Listening Ports

3. Intrusion-Prevention System
    - ❑ Install System
    - ❑ Monitor and Jail Potential Malicious IPs
    - ❑ Whitelist SysAdmin and DevOps IPs

4. Denial of Service/Syn Flood Attack
    - ❑ DoS/Syn protection

5. Secure Shared Memory (if possible)
    - ❑ Secure Shared Memory (if possible)

6. Secure Private Key
    - ❑ Secure a copy of the private and public keys
    - ❑ Implement Hidden Folder
    - ❑ Deploy Decoy (Honey Pot)
    - ❑ Log all access to real keys and decoy
    - ❑ Automate reporting of logs

General Node Setup

The Official Setup should be followed in conjunction with the security measures that are outlined in this document. It is recommended the the SysAdmin or DevOps team first complete #1 shown on the Security Checklist before setting up the validator node. Multifactor authentication is probably the only one that can be delayed from that section. All other security features can be implemented after the validator node has been synced and is up and running.

It is possible to implement all features before the set up the node with the exception of private key protection. In this case, the SysAdmin or DevOps team will need to accept that if the node setup requires a full wipe to restart building, all the security work will need to be redone. Another option is to take a snapshot of the server after security has been implemented and restore to the snapshot if the node needs rebuilding.

Implementing Security Features

This section will describe the security features implemented. A brief description of why the features should be implemented will be given. In addition, some scripts and screenshots are given to aid a SysAdmin or DevOps Team.

Login and root user

Remote logins are the easiest way to try and attack a server. Remote login passwords are generally easy to guess or can be brute forced. In this particular setup we will eliminate this risk by forcing any remote user's to have a private ssh key, changing the ssh service port, and by using 2FA. We will also disable the root account. Any commands needing root should require sudo by a non-root user so that the commands are logged and can be analyzed.

Change The SSH Port

Most hackers will check the default port 22 for ssh. We can discourage many hackers by simply changing the port. Additionally, port knocking or a tarpit can be installed for added protection. In the sshd_config file simply uncomment and change the port configuration to 3987. This port was chosen arbitrarily and can be changed.

```
# in sshd_config
Port 3987
```

Note: Since AWS is being used, you must edit the security group of the instance so that it allows connections to that port from any source that you choose.

Disable ssh root user login

```
# in sshd_config
PermitRootLogin no
```

Disable ssh password authentication

```
# in sshd_config
PasswordAuthentication no
PermitEmptyPasswords no
```

Disable root account

```
# use the -l option to disable the root account
sudo passwd -l root
```

Non-root user login with ssh key

1. Create a root user with a password. The password will not be used to login but should still be created and remembered.
2. Add the ssh public key to the authroized_keys file
3. Finally add the non-root user to the sudo-ers file.

Two-Factor Authentication or Multi-Factor Authentication (2FA or MFA)

1. Install the google authenticator module
2. Secondly, edit the sshd_config file to enable PAM.

```
# in sshd_config
ChallengeResponseAuthentication yes
UsePAM yes
```

3. Then edit the pamd sshd file according to the description below.

```
# in sshd file
#comment out the following line
#@include common-auth
#add the following line to the bottom
auth required pam_google_authenticator.so
```

4. Finally switch to your non-root user and run the google-authenticator It will ask you a few questions and the recommended answers are shown below.
   - Make tokens time-base: yes
     - Update the authenticator file: yes
     - Disallow multiple uses: yes
     - Increase the original generation time limit: no
     - Enable rate-limiting: yes

   A QR code should have already been generated. You can use any authentication app on your phone to set up as the authenticator.

   Note: *Recommendations for apps are DUO Mobile or Google's authenticator app. But other apps are acceptable if approved by the entity security team.*

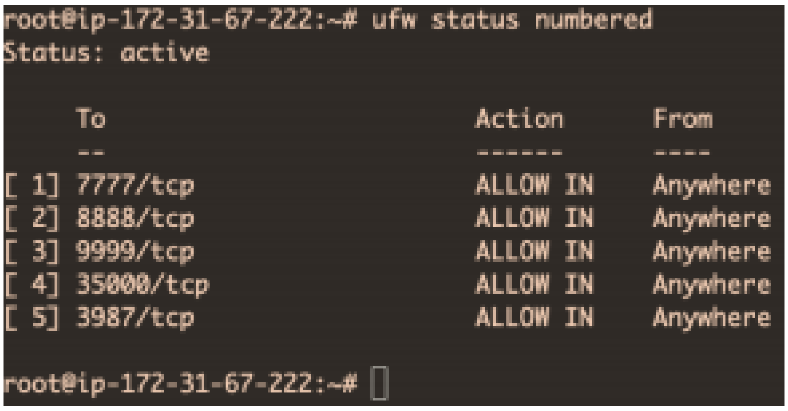
   DO NOT close your current session. Simply open a new session and test your non-root user login. Make sure you have to use your ssh key and also that it asks for your 2FA verification key. If login doesn't work, make sure you review all previous steps. Closing your session prematurely could result in lockout.

Firewall

Blocking all unused ports and only opening the ports that are needed reduces the number of attack surfaces on your system. You can use any firewall package to manage the firewall and the following settings can be used as a guide to set up your own package. The Casper Validator Node is set up with the following configurations using UFW.

Block all unused ports

By default, deny all incoming traffic and allow all outgoing traffic. Then open up any ports needed (don't forget about the new ssh port)



```
root@ip-172-31-67-222:~# ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 7777/tcp                   ALLOW IN    Anywhere
[ 2] 8888/tcp                   ALLOW IN    Anywhere
[ 3] 9999/tcp                   ALLOW IN    Anywhere
[ 4] 35000/tcp                  ALLOW IN    Anywhere
[ 5] 3987/tcp                   ALLOW IN    Anywhere

root@ip-172-31-67-222:~# 
```

Verify Listening Ports

You can use many different tools to verify listening ports. The following script is given as a starting point to check the existing configurations.

```
# run on validator node
netstat -tulpn
```

Intrusion-Prevention System

In the Casper Validator Node that was setup, Fail2Ban was used as the Intrusion-Prevention system. It monitors log files and searches for patterns that correspond to failed login attempts. If a threshold is met, it will block any possible malicious IPs. To use Fail2Ban follow the steps outlined below.

1. Install Fail2Ban

2. Edit the jail.local file by adding the lines below

```
# in jail.local add lines at the bottom of the file
   [sshd]
  enabled = true
  port = <22 or your random port number>
  filter = sshd
  logpath = /var/log/auth.log
  maxretry = 3
  # whitelisted IP addresses
  ignoreip = <whitelisted IP address 1, whitelisted
```

```
IP address 2>
```

3. Whitelist any SysAdmin and DevOps IPs by editing the jail.local file. The ignoreip parameter accepts IP addresses, IP ranges or DNS hosts that you can specify to be allowed to connect. Separate all values by spaces as shown in the example below

```
# Exampleignoreip = 192.168.1.0/24 127.0.0.1/8
```

Note: *Fail2Ban is an open source intrusion prevention software framework that protects computer servers from brute-force attacks. Written in the Python programming language, it is able to run on POSIX systems that have an interface to a packet-control system or firewall installed locally, for example, iptables or TCP Wrapper. Other intrusion prevention software is acceptable for the purpose of protecting the staking node.*

Denial of Service Attack/Syn Flood Attack

This novel attack is meant to lock up your servers communications by flooding your ports with connections. This novel attack is an easy attack for anyone who wants your service to stop. In the case of being a Casper Validator, you must keep your service running 24/7 and operators can potentially be punished for not being available. We can stop these attacks simply by limiting the number of connections any single IP can have to a port. This can be done using iptables by using the script below.

```
# run on validator node
iptables -I INPUT -p tcp -m tcp \
--dport <PORT#> \
--tcp-flags FIN,SYN,RST,ACK SYN -m connlimit \
--connlimit-above <limit#> \
--connlimit-mask 32 \
--connlimit-saddr -j REJECT \
--reject-with tcp-reset
```

In the Casper Validator Node, a limit of 5 connections was set for all ports that are left open. DDoS attacks are a different kind of attack and are discussed in a later section.

Secure Shared Memory

Shared memory is an efficient means of passing data between running programs. By default this memory is mounted as read/write. Most of these exploits target specific running services but this still introduces an attack surface that needs to be secured.

WARNING: The casper node setup is fairly new. As of the time shown on the last update to this document, securing shared memory does not affect the validator node's services. This means that in the Casper Node's current state of development, shared memory does not need to be mounted in read/write mode. If this changes, this section will be updated to allow shared memory with read/write access but without permissions to execute programs, change the UID of any running programs, or to create block devices or character devices in the namespace.

As of the time of the last update, the following edit to fstab will mount Secure Shared Memory in read only mode. Reboot the server after modification.

```
#in fstab - add to bottom
tmpfs /run/shm tmpfs ro,noexec,nosuid 0 0
```

Secure Private Key

This section outlines methods that can be used to secure the private keys. It's important to keep the private key secure because it proves your identity and ownership of the public addresses. Unfortunately, the private key that's created must remain on the node, however, there are many steps we can take to mitigate the risks. The first steps have already been taken since the only users allowed to log in must have 3 things to actually log in. They must have the private ssh key of any user allowed to log in, the password to that key, and the Two-Factor Authentication password which changes every 30 seconds.

In addition to this security, we can implement security features just in case a malicious user somehow gains access to the server. To outline the basic steps, the operator first moves the keys from the default directory to a hidden folder that is still accessible by the config.toml file. Then, a decoy is deployed in the default location of the keys. While this could fool some users, it will not fool everyone so, the operator will also set up automated monitoring to notify Admins if there was any access to the decoy keys or the real keys.

Backup private and public keys

A copy of the private and public keys as well as the public hex should be backed up in cold storage. This can be any system or thumb drive that is never connected to the network. Multiple copies are recommended.

Implement Hidden Directory

You can move keys to a hidden directory anywhere on the machine but if the casper-node service cannot find your private keys, the service will fail to start. You must edit the config.toml file with the correct path to your private key as shown below.

```
# in the config.toml file
secret_key_path = '<new path to key>'
```

The keys can be located in any directory owned by root. You can create a hidden directory by adding a '.' in front of it's name similar to the '.ssh' directory that you may already be familiar with. For example

```
     # make a hidden directory named ".hidden" in
/root/etc directory
     mkdir /root/etc/.hidden
```

While this method doesn't necessarily hide the folder from an adversary who gained root access, it does move the keys from the default location in which most hackers might expect. This process in combination with the decoy that will be set in the default location, and also the monitoring of the decoy and real keys, gives us a good chance to detect any attempt to access the secret key.

Deploy a Decoy (Honey Pot)

In the default location, the validator_keys folder should remain in place. Fake keys can be generated by just generating keys again on the server. These keys will never be used but they will look real to an intruder. By leaving the keys in the default location, an attacker may just grab the fake keys and leave. After setting up the automated Logs in the next section, we will be able to detect if the decoy or the real keys have been accessed.

Log All Access to Decoy and Real Keys

A regular attacker may just take the decoy keys, but a creative attacker may know how to check where the keys actually are if they gain access to your system. The good news is that the keys actually never really need to be accessed once they are created, moved into place, and the node is running. This means that there should be no reason to access those files by any user. It will be accessed by the casper-node service though so those access reports can be ignored.

Because of this special case, it is recommended to use inotify-tools to monitor access to monitor the decoy and real directories and their contents. Specifically, we will use the inotifywait to recursively watch both directories and their contents for specific events. These events include open, close, access, modify, move, delete, and create. Again, these files don't need to be accessed by Admins so it won't blow up the logs. An example of inotify for this purpose is shown below

```
# bash script to run inotify tools and output reports to a
log inotifywait -r -d /etc/casper/validator_keys \
-e access \
-e open \
-e modify \
-e move \
-o /var/log/access.log
```

Not all the listed events need to be monitored. You can simply monitor the access event and it will be just as effective. There may be some changes that are needed in order to make sure the event is not detected for the casper-node service.

Automate Reporting of Logs

For the logs in the previous step to be useful, it must be monitored. There are a few options depending on the level of risk you want to take. In the case of this particular Casper Validator node, A custom systemd service was built to send an email if it detects that the log has been updated. A sample of that script is shown below. The node uses mailutils in a custom script to send a formatted email with the log attached. This can be done with any email client and the basics of the monitoring script is shown below.

```
#!/bin/bash
cur_line_count="$(wc -l myfile.txt)"
while true
do
      new_line_count="$(wc -l myfile.txt)"
   if [ "$cur_line_count" != "$new_line_count" ]
      then
             ./email_log.sh
      fi
      cur_line_count="$new_line_count"
      sleep 5
done
```

Note: If the event access is being monitored by you may get a log everytime casper-node accesses your key. You can prevent this by detecting other events instead.

Distributed Denial of Service Attack (DDoS)

The goal of this type of attack is similar to the DoS attack described earlier. This attack is designed to stop your services from running by flooding communications. A DDoS is a very serious specific target attack and can be difficult to mitigate even if detected. Usually, you will need a DevOps Team that knows how to isolate where all the traffic is coming from and then block their IPs. If you are lucky, all the traffic is at least originating from the same subnet. Since this is an AWS server, AWS Shield Standard is available to all AWS customers at no extra cost and protects from some of the most common attacks like a DDoS. If AWS Shield Standard is not enough to meet the entity security needs as established in the information security assessment and risk appetite, more options are given below.

- AWS advanced - just upgrade to a paid plan with AWS.
- Implement a custom DDoS detection with iptables and monitor with a team (internal or outsourced).
- Cloud Flare or any other out of the box solutions available in the market.

Recommendation on DDoS

DDoS attacks are complex target specific attacks. A basic solution is all that's needed for any server since there is no reason to suspect a DDoS attack. There is a very small chance of a DDoS attack. Still, it's important to have some security features handling this. It's recommended that the AWS Shield standard be used to mitigate these attacks. If a DDoS attack occurs, AWS Shield will report it and the previous section on DDoS attacks can be used to map out a way forward. It may very well be the case that the AWS Shield standard was effective enough to mitigate the attack.

***Note:** For the assessment purpose AWS Shield was used in the POC (Proof of concept) since AWS was the recommended environment for the Casper Node. AWS Shield is a managed Distributed Denial of Service (DDoS) protection service that safeguards applications running on AWS. Any product that performs similar functions will suffice to mitigate such risks.*

Summary

This document provides an overview of the base practices and security features that should be considered when deploying a Casper Validator Node. One of the most concerning security issues is that the private key needs to remain on the server. This can be mitigated by placing the keys in a hidden folder, setting up decoy keys, and by monitoring both the decoy and real keys. A basic checklist is given and if the checklist is fully utilized, the Casper Validator Node will be Secure with a very high degree of confidence.