Configure Nginx Reverse Proxy for VPN-Less Access to Cisco Finesse (12.6 ES03)

Contents

Introduction

This document describes how to use a reverse proxy to access the Cisco Finesse desktop without connecting to a VPN based on 12.6 ES03 versions of Cisco Finesse, Cisco Unified Intelligence Center (CUIC), and Cisco Identity Service (IdS).

Note: Nginx installation and configuration is not supported by Cisco. Queries in regards to this subject can be discussed on the Cisco community forums.

Note: For ES03 deployments of VPN-Less, see the readme of the individual components in order to plan the upgrades and check compatibility restrictions. Cisco Finesse 12.6 ES03 Readme, CUIC / IdS 12.6 ES03 Readme

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Cisco Unified Contact Center Enterprise (UCCE) Release
- Cisco Finesse
- Linux administration
- Network administration and Linux network administration

Components Used

The information in this document is based on these software and hardware versions:

- Finesse - 12.6 ES03
- CUIC - 12.6 ES03
- IdS - 12.6 ES03
- UCCE/Hosted Collaboration Solution (HCS) for Contact Center (CC) - 11.6 or later
- Packaged Contact Center Enterprise (PCCE) - 12.5 or later

  Note : PCCE/UCCE 2k deployments will need to be on 12.6 version of CCE due to LD/CUIC co-resident deployment

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Note: The configuration provided in this document has been configured, hardened, and load tested with Nginx reverse proxy (OpenResty) deployed on CentOS 8.0, against a sample 2000 user UCCE deployment. Performance profile reference information is available in this document.

Background Information

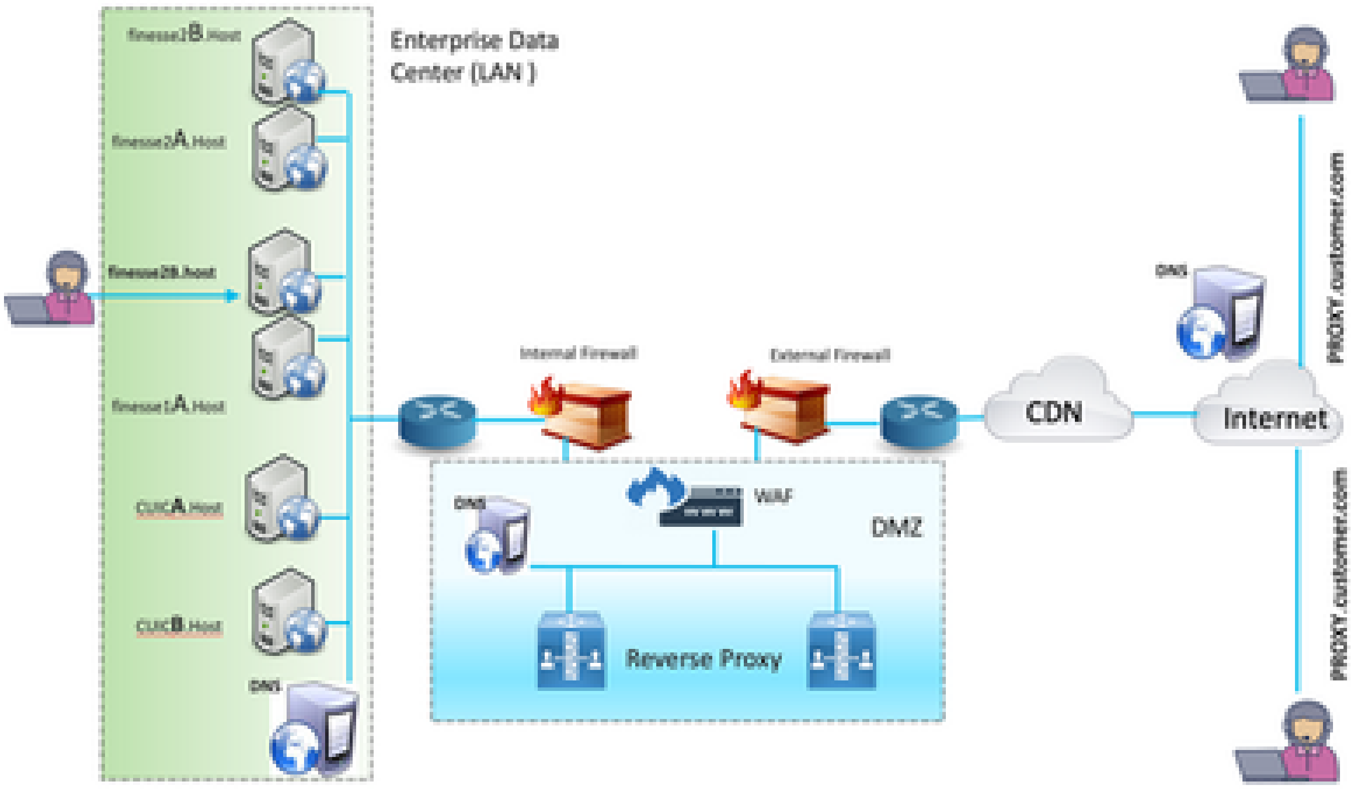
This deployment model is supported for the UCCE/PCCE and HCS for UCCE solutions.

Deployment of a reverse proxy is supported (available from 12.6 ES01) as an option to access the Cisco Finesse desktop without connecting to a VPN. This feature provides the flexibility for agents to access the Finesse desktop from anywhere through the Internet.

In order to enable this feature, a reverse proxy pair must be deployed in the Demilitarized Zone (DMZ).

Media access remains unchanged in reverse proxy deployments. To connect to the media, agents can use Cisco Jabber over Mobile and Remote Access solution (MRA) or the Mobile Agent capability of UCCE with a Public Switched Telephone Network (PSTN) or mobile endpoint. This diagram shows how the network deployment will look like when you access two Finesse clusters and two CUIC nodes through a single high availability (HA) pair of reverse proxy nodes.

Concurrent access from agents on the Internet and agents that connect from the LAN is supported as shown in this image.



Note: See the feature guide for third-party proxy selection criteria in place of Nginx to support this deployment.

- UCCE 12.6 Feature Guide - Provides a feature overview, design, as well as **configuration details** for the VPN-Less Feature.
- UCCE 12.6 Security Guide - Provides security configuration guidelines for the reverse proxy

deployment.

It is recommended to review the VPN-Less section of the features guide and the security guide before you read this document.

Changes in ES03

- New Features
  - Finesse supervisor capabilities are now supported via reverse proxy.
  - CUIC RealTime and Historical reports are now supported via Finesse gadgets in a proxied environment.
  - Authentication for all requests / communications - requires Lua support
    - All Finesse / CUIC / IM & Presence (IM&P) requests are authenticated at the proxy before being allowed to enter the datacenter.
    - Websocket and Live data socketIO connections are also restricted and allowed only from clients which have successfully made a secured request to Finesse.
    - Brute force attack sensing and logging at the proxy, which can be used with Fail2Ban to block malicious IP addresses.
- Security Enhancements for reverse proxy configuration - requires Lua support
  - Mutual Transport Layer Security (TLS) authentication between reverse proxy and upstream components (Finesse/IdS/CUIC/Livedata).
  - SeLinux settings.
  - Enable mutual Secure Sockets Layer (SSL) trust verification for proxy and component server requests.
- Enhanced security for the proxy configuration to prevent Denial-of-Service (DoS) / Distributed Denial-of-Service (DDoS) attacks - requires Lua support
  - Enhanced Nginx request rate limits for various parts of the system.
  - Rate limits for IpTables.
  - Verification of static resource requests before requesting the upstream component server.
  - Lighter and cacheable unauthenticated pages which do not hit the upstream component server.
- Miscellaneous other features - requires Lua support
  - Auto sensing Cross-Origin Resource Sharing (CORS) responses provided from the proxy to aid automatic configuration and to improve the performance
- Defect Fixes relating to VPN-Less
  - CSCwa26057

CSCwa26057

" />- Multiple Certificates offered to agent during finesse desktop login

- CSCwa24471

- Finesse login page does not show SSO Agent FQDN name

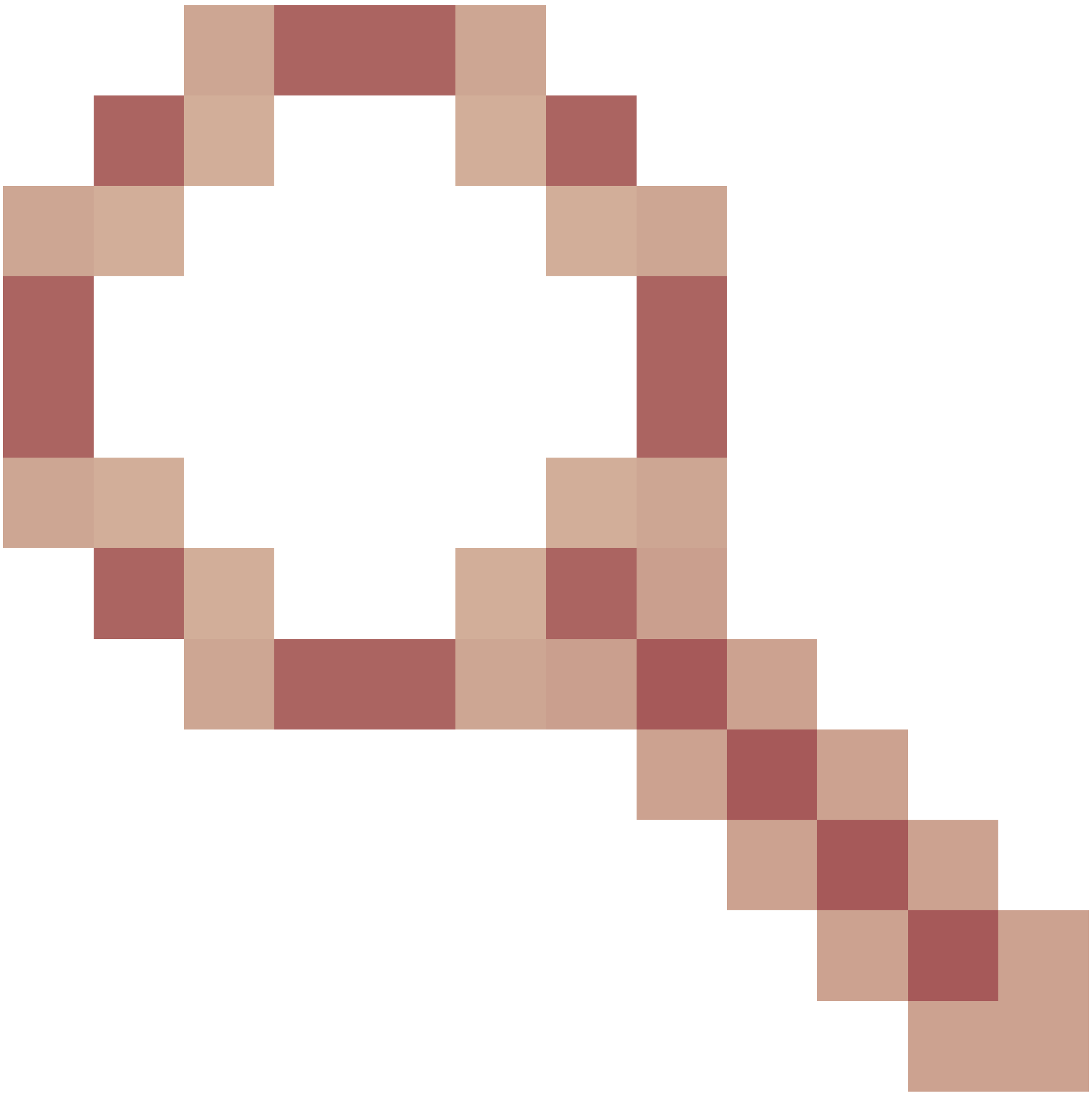
- CSCwa24519

: Webproxy service fails to restart if reverse proxy hostname is not resolvable from component

- CSCwa23252

: Proxy finesse trust is broken when depth is more than one for CA certs chain

- CSCwa46459


log4j zero day vulnerability exposed in webservice

Upgrade Notes for ES01-Based VPN-Less Configurations

- ES03 configuration requires Nginx installation with Lua support.
- Certificate Requirements
  - Cisco Finesse, CUIC, and IdS will require the Nginx / OpenResty host certificate to be added to the Tomcat trust store and a restart done, before the Nginx ES02 configuration will be able to successfully connect to the upstream server.
  - Cisco Finesse, CUIC, and IdS upstream server certificates need to be configured in the Nginx server to use the ES03-based configuration.

Note: It is recommended to remove the existing ES01-based Nginx configuration before you install the ES03 Nginx configurations.

Note: ES03 configuration scripts also require the corresponding ES03 COP installation in Cisco Finesse, CUIC, and IdS.

Authentication

Finesse 12.6 ES03 introduces authentication at the proxy. Authentication is supported for Single Sign On (SSO) and non-SSO deployments.

Authentication is enforced for all requests and protocols that are accepted at the proxy before they are forwarded to the upstream component servers, where the authentication enforced by the component servers locally take place too. All authentication uses the common Finesse login credentials to authenticate the requests.

Persistent connections, such as websockets which rely on application protocols like Extensible Messaging and Presence Protocol (XMPP) for authentication and post connection, are authenticated at the proxy by validating the IP address from which a successful application authentication has been made prior to establishing the socket connection.

Non-SSO Authentication

Non-SSO authentication does not require any extra configurations and will work with out of the box Nginx configuration scripts once the required script replacements are made. Authentication relies on the user name and password used to log in to Finesse. Access to all endpoints will be validated with Finesse authentication services.

The list of valid users is cached at the proxy locally (updates the cache every 15 minutes), which is used to validate the user in a request. User credentials are validated by forwarding the request to the configured Finesse URI and thereafter the credential hash is cached locally (cached 15 minutes) to authenticate new requests locally. If there is any change to the username or password, it will be taken into effect only after 15 minutes.

SSO Authentication

SSO authentication requires that the administrator configure the IdS token encryption key at the Nginx server within the configuration file. The IdS token encryption key can be obtained from the IdS server with the **show ids secret** CLI command. They key has to be configured as part of one of the #Must-change replacements that the administrator has to perform in the scripts before the SSO auth can work.

Refer to the SSO user guide for the IdS SAML configurations to be performed for the proxy resolution to work for IdS.

Once SSO authentication is configured, a valid pair of tokens can be used to access any of the endpoints in the system. The proxy configuration validates the credentials by intercepting the token retrieval requests made to IdS or by decrypting valid tokens and thereafter caching them locally for further validations.

Authentication for Websocket Connections

Websocket connections cannot be authenticated with the standard authorization header, as custom headers are not supported by native websocket implementations in the browser. Application level authentication protocols, where the authentication information contained in the payload does not prevent websocket connection establishment, and hence malicious entities can render DOS or DDOS attacks just by creating innumerable connections to overwhelm the system.

In order to mitigate this possibility, the nginx reverse proxy configurations provided have specific checks to allow websocket connections to be accepted ONLY from those IP addresses which have successfully made an authenticated REST request prior to establishment of the websocket connection. This means that clients which attempt to create websocket connections, before a REST request is issued, will now get an authorization failed error and is not a supported usage scenario.

Brute Force Attack Prevention

Finesse 12.6 ES02 authentication scripts actively prevent brute force attacks which can be used to guess the user password. It does this by blocking the IP address used to access the service, after a certain number of failed attempts in a short time. These requests will be rejected by **418 client error**. The details of the blocked IP addresses can be accessed from the files **<nginx-install-directory>/logs/blocking.log** and **<nginx-install-directory>/logs/error.log.**

The number of failed requests, time interval, and blocking duration is configurable. Configurations are present in the **<nginx-install-directory>/conf/conf.d/maps.conf** file.

```
## These two constants indicate five auth failures from a client can be allowed in thirty seconds.
## if  the threshold is crossed,client ip will be blocked.
map $host $auth_failure_threshold_for_lock {
    ## Must-change Replace below two parameters as per requirement
    default 5 ;
}

map $host $auth_failure_counting_window_secs {
    ## Must-change Replace below two parameters as per requirement
    default 30;
}

## This indicates duration of blocking a client to avoid brute force attack
map $host $ip_blocking_duration {
    ## Must-change Replace below parameter as per requirement
    default 1800;
}
```

Logging

To find the IP addresses that are blocked, run the following commands from the directory **<nginx-install-directory>/logs.**

```
grep "will be blocked for" blocking.log
grep "IP is already blocked." error.log
```

```
2021/10/29 17:30:59 [emerg] 1181750#1181750: *19 [lua] block_unauthorized_users.lua:153:
_redirectAndSendError(): 10.68.218.190 will be blocked for 30 minutes for exceeding retry limit.,
client: 10.68.218.190, server: saproxy.cisco.com, request:
"GET /finesse/api/SystemInfo?nocache=1636456574482 HTTP/2.0", host: "saproxy.cisco.com:8445",
referrer: "https://saproxy.cisco.com:8445/desktop/container/?locale=en_US&"

2021/10/29 19:21:00 [error] 943068#943068: *43 [lua] block_unauthorized_users.lua:53: 10.70.235.30 ::
IP is already blocked..., client: 10.70.235.30, server: saproxy.cisco.com, request:
"GET /finesse/api/SystemInfo?nocache=1635591686497 HTTP/2.0", host: "saproxy.cisco.com:8445",
referrer: "https://saproxy.cisco.com:8445/desktop/container/?locale=en_US"
```

It is recommended that customers integrate with Fail2ban or similar to add the ban to the IPtable/firewall rules.

Install and Configure Fail2ban

Fail2ban scans log files and ban IPs that show the malicious signs - too many password failures, seeking for exploits, etc. Generally, Fail2Ban is then used to update firewall rules to reject the IP addresses for a specified amount of time, although any arbitrary other action (e.g. sending an email) could also be configured. For more information, visit https://www.fail2ban.org/.

Fail2ban can be configured to monitor the blocking.log to identify the IP addresses that are blocked by Nginx on detecting bruteforce attacks, and ban them for a configurable duration. Steps to install and configure fail2ban on a CentOS reverseproxy are as follows:

1. Install Fail2ban using yum.

```
yum update && yum install epel-release
yum install fail2ban
```

2. Create a local jail.

Jail configurations allow the administrator to configure various properties like the ports that are to be banned from being accessed by any blocked IP address, the duration for which the IP address stays blocked, the filter configuration used for identifying the blocked IP address from the log file monitored, etc. Steps to add a custom configuration for banning IP addresses that are blocked from accessing the upstream servers are as follows:

2.1. Go to Fail2ban installation directory (in this example /etc/fail2ban)

```
cd /etc/fail2ban
```

2.2. Make a copy of **jail.conf** into jail.local to keep the local changes isolated.

```
cp jail.conf jail.local
```

2.3. Add these jail configurations to the end of the file **jail.local**, and substitute the ports in the template with the actual ones. Update ban time configurations as required.

```
# Jail configurations for HTTP connections.
[finesse-http-auth]
enabled = true
# The ports to be blocked. Add any additional ports.
port = http,https,<finesse-ports>,<cuic-ports>,<any-other-ports-to-be-blocked>
# Path to nginx blocking logs.
```

```
logpath = /usr/local/openresty/nginx/logs/blocking.log
# The filter configuration.
filter = finesseban
# Block the IP from accessing the port, once the IP is blocked by lua.
maxretry= 1
# Duration for retry set to 3 mins. Doesn't count as the maxretry is 1
findtime= 180
# Lock time is set to 3 mins. Change as per requirements.
bantime = 180
```

3. Configure a filter.

A filter tells Fail2ban what to look for in the logs to identify the host to be banned. The steps to create a filter is as below:

3.1. Create **filter.d/finesseban.conf**.

```
touch filter.d/finesseban.conf
```

3.2. Add these lines into the file **filter.d/finesseban.conf**.

```
[Definition]
# The regex match that would cause blocking of the host.
failregex = <HOST> will be blocked for
```

4. Start Fail2ban.

Run this command to start fail2ban.

```
fail2ban-client start
```

Open fail2ban log files and verify that there are no errors. By default, logs for fail2ban go into the file **/var/log/fail2ban.log**.

Validate Static Resource URLs

All valid endpoints which can be accessed in an unauthenticated fashion are actively kept track of in the ES03 scripts.

Requests to these unauthenticated paths are actively rejected, if an invalid URI is requested, without sending these requests to the upstream server.

Caching CORS Headers

When the first options request is successful, the response headers **access-control-allow-headers, access-**

control-allow-origin, **access-control-allow-methods, access-control-expose-headers**, and **access-control-allow-credentials** are cached at the proxy for five minutes. These headers are cached for each respective upstream server.

Configure

This document describes the configuration of Nginx as the reverse proxy to be used to enable Finesse VPN-Less access. The UCCE solution component, proxy and OS versions used to verify the instructions provided are provided. The relevant instructions have to be adapted to the OS/proxy of your choice.

- Nginx version used - OpenResty 1.19.9.1
- OS used for configuration - CentOS 8.0

Note: The Nginx configuration described can be downloaded from the Finesse Release 12.6(1)ES3 software download page.

Configure solution components for VPN Less access

Post configuring the proxy, do configure the solution components (Finesse/ CUIC / IdS) for VPN Less access with the planned hostname and IP of the proxy/services used to access the solution with these commands.

```
utils system reverse-proxy allowed-hosts add
utils system reverse-proxy config-uri <uri> add
```

The details of these commands can be found in the UCCE 12.6 Feature Guide and should be referred to before you use this document.

Install OpenResty as a Reverse Proxy in DMZ

This section details the OpenResty based proxy installation steps. The reverse proxy is typically configured as a dedicated device in the network demilitarized zone (DMZ) as shown in the deployment diagram that was mentioned previously.

1. Install the **OS of your choice** with the required hardware specification. Kernel and IPv4 parameter tweaks might differ depending on the OS selected and users are advised to reverify these aspects if the chosen OS version is different.
2. Configure two network interfaces. One interface will be required for public access from the Internet clients and another to communicate with the servers in the internal network.
3. Install OpenResty.

Any flavors of Nginx can be used for this purpose, as long as they are based on Nginx 1.19+ and support Lua:

- Nginx Plus
- Nginx Open Source (Nginx open source will need to be compiled along with OpenResty-based Lua modules for it to be used)
- OpenResty
- GetPageSpeed Extras

Note: The configuration provided has been tested with OpenResty 1.19 and is expected to work with other distributions with only minor updates, if any.

OpenResty Installation

1. Install OpenResty. See OpenResty Linux Packages. As part of the OpenResty installation, Nginx will be installed in this location and add the OpenResty path to the **PATH** variable by adding in the **~/.bashrc** file.

   ```
   export PATH=/usr/local/openresty/bin:$PATH
   ```

2. Start / stop Nginx.
   - In order to start Nginx, enter **openresty**.
   - In order to stop Nginx, enter **openresty -s stop**.

Configure Nginx

The configuration is explained for an OpenResty-based Nginx installation. The default dirctories for OpenResty are:

- <nginx-install-directory> = /usr/local/openresty/nginx
- <Openresty-install-directory> = /usr/local/openresty

1. Download and extract the file from the Finesse Release 12.6(1)ES03 software download page (12.6-ES03-reverse-proxy-config.zip) that contains the reverse proxy configuration for Nginx.
2. Copy **nginx.conf**, **nginx/conf.d/**, and **nginx/html/** from the extracted reverse proxy configuration directory to **<nginx-install-directory>/conf**, **<nginx-install-directory>/conf/conf.d/**, and **<nginx-install-directory>/html/** respectively.
3. Copy the **nginx/lua** directory from the extracted reverse proxy configuration directory inside the **<nginx-install-directory>.**
4. Copy the contents of **lualib** to **<Openresty-install-directory>/lualib/resty**.
5. Configure nginx log rotation by copying the **nginx/logrotate/saproxy** file to the **<nginx-install-directory>/logrotate/** folder. Modify the file contents to point to the correct log directories if Nginx defaults are not used.
6. Nginx must be run with a dedicated non-privileged service account, which must be locked and have an invalid shell (or as applicable for the chosen OS).
7. Find the "**Must-change**" string in the files under the extracted folders named **html** and **conf.d** and replace the indicated values with appropriate entries.
8. Ensure that all mandatory replacements are done, which are described with the **Must-change** comments in the config files.
9. Make sure the cache directories configured for CUIC and Finesse are created under **<nginx-install-directory>/cache** along with these temporary directories.
   - <nginx-install-directory>/cache/client_temp
   - <nginx-install-directory>/cache/proxy_temp

Note: The configuration provided is for a sample 2000 deployment and has to be expanded appropriately for a larger deployment.

Configure the Nginx Cache

By default, the proxy cache paths are stored in the file system. We recommend changing them to in-memory drives by creating a cache location in tmpfs as shown here.

1. Create directories for the different proxy cache paths under /home.

   As an example, these directories must be created for the primary Finesse. The same steps should be followed for the secondary Finesse and CUIC servers.

   ```
   mkdir -p /home/primaryFinesse/rest
   mkdir -p /home/primaryFinesse/desktop
   mkdir -p /home/primaryFinesse/shindig
   mkdir -p /home/primaryFinesse/openfire
   mkdir -p /home/primaryCUIC/cuic
   mkdir -p /home/primaryCUIC/cuicdoc
   mkdir -p /home/client_temp
   mkdir -p /home/proxy_temp
   ```

   ```
   echo "tmpfs /home/primaryFinesse/rest tmpfs size=1510M,rw,auto,noexec,nodev,nosuid,gid=root,uid=root,mode=1700 0 0" >>
   /etc/fstab echo "tmpfs /home/primaryFinesse/desktop tmpfs size=20M,rw,auto,noexec,nodev,nosuid,gid=root,uid=root,mode=1700 0 0"
   >>
   /etc/fstab echo "tmpfs /home/primaryFinesse/shindig tmpfs size=500M,rw,auto,noexec,nodev,nosuid,gid=root,uid=root,mode=1700 0 0"
   >>
   /etc/fstab echo "tmpfs /home/primaryFinesse/openfire tmpfs size=10M,rw,auto,noexec,nodev,nosuid,gid=root,uid=root,mode=1700 0 0"
   >>
   /etc/fstab echo "tmpfs /home/primaryCUIC/cuic tmpfs size=100M,rw,auto,noexec,nodev,nosuid,gid=root,uid=root,mode=1700 0 0" >>
   /etc/fstab echo "tmpfs /home/primaryCUIC/cuicdoc tmpfs size=100M,rw,auto,noexec,nodev,nosuid,gid=root,uid=root,mode=1700 0 0"
   >>
   /etc/fstab echo "tmpfs /home/client_temp tmpfs size=2048M,rw,auto,noexec,nodev,nosuid,gid=root,uid=root,mode=1700 0 0" >>
   /etc/fstab echo "tmpfs /home/proxy_temp tmpfs size=2048M,rw,auto,noexec,nodev,nosuid,gid=root,uid=root,mode=1700 0 0" >>
   /etc/fstab
   ```

   Note: Increase the client and proxy_temp caches by 1 GB for each new Finesse cluster added to the configuration.

2. Mount the new mount points with the command **mount -av.**
3. Validate the filesystem has mounted the new mount points with the **df -h** command.
4. Change the proxy_cache_path locations in the Finesse and CUIC cache configuration files.

   For example, to change the paths for the Finesse primary, go to **<nginx-install-directory>conf/conf.d/finesse/caches** and change the existing cache location **/usr/local/openresty/nginx/cache/finesse25/** to the newly created filesystem location **/home/primaryFinesse.**

   ```
   ##Must-change /usr/local/openresty/nginx/cache/finesse25 location would change depending on folder extraction proxy_cache_path
   /home/primaryFinesse/desktop levels=1:2 use_temp_path=on keys_zone=desktop_cache_fin25:10m max_size=15m inactive=3y
   use_temp_path=off; proxy_cache_path /home/primaryFinesse/shindig levels=1:2 use_temp_path=on
   keys_zone=shindig_cache_fin25:10m max_size=500m inactive=3y use_temp_path=off; proxy_cache_path
   /home/primaryFinesse/openfire levels=1:2 use_temp_path=on keys_zone=openfire_cache_fin25:10m max_size=10m inactive=3y
   use_temp_path=off; proxy_cache_path /home/primaryFinesse/rest levels=1:2 use_temp_path=on keys_zone=rest_cache_fin25:10m
   max_size=1500m inactive=40m use_temp_path=off;
   ```

5. Follow the same steps for the Finesse secondary and CUIC servers.

Note: Ensure that sum of all the tmpfs drive sizings created in all the previous steps is added to the final memory sizing for the deployment, since these drives are memory blocks configured to look like disks to the application and consume as much memory space.

Configure SSL Certificates

Use Self-Signed Certificates - Test Deployments

Self signed certificates should only be used until the reverse proxy is ready to be rolled out into production. On a production deployment, use only a Certificate Authority (CA) signed certificate.

1. Generate Nginx certificates for SSL folder content. Before you generate certificates, you need to create a folder called **ssl** under **/usr/local/openresty/nginx**. You need to generate two certificates with the help of these commands (one for **<reverseproxy_primary_fqdn>** and another for <**reverseproxy_secondary_fqdn**>).
   a. **sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /usr/local/openresty/nginx/ssl/nginx.key -out /usr/local/openresty/nginx/ssl/nginx.crt** (pass host name as : <reverseproxy_primary_fqdn>)
   b. **sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /usr/local/openresty/nginx/ssl/nginxnode2.key -out /usr/local/openresty/nginx/ssl/nginxnode2.crt** (pass host name as :<reverseproxy_secondary_fqdn>)
   c. Ensure that the certificate path is **/usr/local/openresty/nginx/ssl/nginx.crt** and **/usr/local/openresty/nginx/ssl/nginxnode2.crt**, since these are already configured in Finesse Nginx configuration files.
2. Change the permission of the private key **400 (r--------)**.
3. Configure the firewall/**iptables** on the reverse proxy to enable communication from the firewall to correspond to the ports on which the Nginx server has been configured to listen.
4. Add the IP address and hostname of Finesse, IdS, and CUIC under the **/etc/hosts** entry on the reverse proxy server.
5. Refer to the solution features guide for the configurations to be performed on the component servers to configure the Nginx host as a reverse proxy.

Note: The configuration provided is for a sample 2000 deployment and has to be expanded appropriately for a larger deployment.

Use CA Signed Certificate - Production Deployments

A CA-signed certificate can be installed on the reverse proxy with these steps:

1. Generate the certificate signing request (CSR).

   In order to generate the CSR and private key, enter**openssl req -new -newkey rsa:4096 -keyout nginx.key -out nginx.csr**after you log into the proxy. Follow the prompt, and provide the details. This generates the CSR (nginx.csr in the example) and the RSA private key (nginx.key in the example) of strength 4096 bits.

   For example:

   [root@reverseproxyhost.companyname.com ssl]# openssl req -new -newkey rsa:4096 -keyout nginx.key -out nginx.csr Generating a RSA private key .....+++++ ....+++++ writing new private key to 'nginx.key' Enter PEM pass phrase:passphrase Verifying - Enter PEM pass phrase:passphrase ----- 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 Name (2 letter code) [XX]:US State or Province Name (full name) []:CA Locality

```
Name (eg, city) [Default City]:Orange County Organization Name (eg, company) [Default Company Ltd]:CompanyName
Organizational Unit Name (eg, section) []:BusinessUnit Common Name (eg, your name or your server's hostname)
[]:reverseproxyhostname.companydomain.com Email Address []:john.doe@comapnydomain.com Please enter the following 'extra'
attributes to be sent with your certificate request A challenge password []:challengePWD An optional company name []:CompanyName
```

Write down the PEM passphrase, as this will be used to decrypt the private key during the deployment.

2. Obtain the signed certificate from the CA.

   Send the CSR to the certificate authority and obtain the signed certificate.

   Note: If the certificate received from the CA is not a certificate chain containing all the respective certificates, compose all the relevant certificates into a single certificate chain file.

3. Deploy the certificate and key.

   Decrypt the key generated earlier as part of the first step with the**openssl rsa -in nginx.key -out nginx_decrypted.key**command. Place the CA signed certificate and the decrypted key inside the folder /usr/local/openresty/nginx/ssl in the reverse proxy machine. Update/add SSL configurations related to the certificate in the Nginx configurations in the configuration file /usr/local/openresty/nginx/conf/conf.d/ssl/ssl.conf.

   ```
   ssl_certificate /usr/local/openresty/nginx/ssl/ca_signed_cert.crt; ssl_certificate_key /usr/local/openresty/nginx/ssl/nginx_decrypted.key;
   ```

4. Configure permissions for the certificates.

   Enter **chmod 400 /usr/local/openresty/nginx/ssl/ca_signed_cert.crt**and**chmod 400 /usr/local/openresty/nginx/ssl/nginx_decrypted.key**, so that the certificate has read-only permission and is restricted to the owner.

5. Reload Nginx.

Use Custom Diffie-Hellman Parameter

Create a custom Diffie-Hellman parameter with these commands:

```
 openssl dhparam -out /usr/local/openresty/nginx/ssl/dhparam.pem 2048 chmod 400 /usr/local/openresty/nginx/ssl/dhparam.pem
```

Alter the server configuration to use the new parameters in the file /usr/local/openresty/nginx/conf/conf.d/ssl/ssl.conf:

```
ssl_dhparam /usr/local/openresty/nginx/ssl/dhparam.pem;
```

Ensure OCSP Stapling is Enabled - Certificate Revocation Check

Note: In order to enable this, the server should be using a CA-signed certificate and the server should have access to the CA which signed the certificate.

Add/update this configuration in the file/usr/local/openresty/nginx/conf/conf.d/ssl/ssl.conf:

ssl_stapling on; ssl_stapling_verify on;

Nginx Configuration

The default Nginx configuration file (**/usr/local/openresty/nginx/conf/nginx.conf**) has to be modified to contain these entries to enforce security and provide performance. This content should be used to modify the default configuration file which is created by the Nginx install.

```
# Increasing number of worker processes will not increase the processing the request. The number of wor
# in system CPU. Nginx provides "auto" option to automate this, which will spawn one worker for each CP
worker_processes  auto;

# Process id file location
pid /usr/local/openresty/nginx/logs/nginx.pid;

# Binds each worker process to a separate CPU
worker_cpu_affinity auto;

#Defines the scheduling priority for worker processes. This should be calculated by "nice" command. In 
worker_priority 0;


error_log  /usr/local/openresty/nginx/logs/error.log info;

#user root root;

# current limit on the maximum number of open files by worker processes, keeping 10 times of worker_con

worker_rlimit_nofile 102400;

events {
    multi_accept on;


    # Sets the maximum number of simultaneous connections that can be opened by a worker process.
    # This should not be more the current limit on the maximum number of open files i.e. hard limit of 
    # The appropriate setting depends on the size of the server and the nature of the traffic, and can 
    worker_connections  10240;
    #debug_connection 10.78.95.21
}


http {

    include       mime.types;


    default_type  text/plain;


    ## Must-change Change with DNS resolver ip in deployment
    resolver 192.168.1.3;


    ## Must-change change lua package path to load lua libraries
    lua_package_path  "/usr/local/openresty/lualib/resty/?.lua;/usr/local/openresty/nginx/lua/?.lua;;"
    ## Must-change change proxy_temp folder as per cache directory configurations
    proxy_temp_path /usr/local/openresty/nginx/cache/proxy_temp 1 2 ;
    ## Must-change change client_temp folder as per cache directory configurations
    client_body_temp_path /usr/local/openresty/nginx/cache/client_temp 1 2 ;


    lua_shared_dict userlist 50m;
    lua_shared_dict credentialsstore 100m;
    lua_shared_dict userscount 100k;
    lua_shared_dict clientstorage 100m;
    lua_shared_dict blockingresources 100m;
    lua_shared_dict tokencache_saproxy 10M;
    lua_shared_dict tokencache_saproxy125 10M;
    lua_shared_dict ipstore 10m;
    lua_shared_dict desktopurllist 10m;
    lua_shared_dict desktopurlcount 100k;
    lua_shared_dict thirdpartygadgeturllist 10m;
    lua_shared_dict thirdpartygadgeturlcount 100k;
    lua_shared_dict corsheadersstore 100k;


    init_worker_by_lua_block {
        local UsersListManager = require('users_list_manager')
        local UnauthenticatedDesktopResourcesManager = require("unauthenticated_desktopresources_manage
        local UnauthenticatedResourcesManager = require("unauthenticated_thirdpartyresources_manager")
        --  Must-change Replace saproxy.cisco.com with reverseproxy fqdn

        if  ngx.worker.id() == 0 then
            UsersListManager.getUserList("saproxy.cisco.com", "https://saproxy.cisco.com:8445/finesse/a
            UnauthenticatedDesktopResourcesManager.getDesktopResources("saproxy.cisco.com", "https://sa
            UnauthenticatedResourcesManager.getThirdPartyGadgetResources("saproxy.cisco.com", "https://
        end
    }

    include conf.d/*.conf;

    sendfile        on;

    tcp_nopush     on;

    server_names_hash_bucket_size 512;
```

Configure Reverse Proxy Port

By default, the Nginx configuration listens on port 8445 for Finesse requests. At a time, only one port can be enabled from a reverse proxy to support Finesse requests, for example 8445. If port 443 needs to be supported, then edit the **<nginx-install-directory>conf/conf.d/finesse.conf** file in order to enable listening on 443 and disable listening on 8445.

Configure Mutual TLS Authentication Between Reverse Proxy and Upstream Components

Client SSL certificate authentication for connections from reverse proxy hosts can be enabled on CCBU upstream components CUIC/Finesse/IdS/Livedata via new CVOS CLI option which is

utils system reverse-proxy client-auth enable/disable/status.

By default this is disabled and has to be explicitly enabled by administrator by executing CLI on on each upstream server independently. Once this option is enabled, Cisco Web proxy Service running on upstream host will start authenticating client certificates in TLS handshake for connections originating from trusted reverse proxy hosts added as part of CLI ***utils system reverse-proxy allowed-hosts add <proxy-host>***.

Below is the configuration block for the same in proxy config files namely ***ssl.conf and ssl2.conf***

```
#Must-change /usr/local/openresty/nginx/ssl/nginx.crt change this location accordingly proxy_ssl_certificate
/usr/local/openresty/nginx/ssl/nginx.crt; #Must-change /usr/local/openresty/nginx/ssl/nginx.key change this location accordingly
proxy_ssl_certificate_key /usr/local/openresty/nginx/ssl/nginx.key;
```

SSL certificate used for outbound traffic(proxy to upstream) can be same as ssl certificate configured for inbound traffic(SSL connector for component server blocks). If self signed certificate is used as proxy_ssl_certificate than it has to be uploaded to upstream components(Finesse/IdS/CUIC/Livedata) tomcat trust store for it to be authenticated successfully.

Upstream server certificate validation by reverse proxy is optional and disabled by default. If you wish to achieve full TLS mutual auth between reverse proxy and upstream hosts, below configuration needs to be uncommented out from ssl.conf and ssl2.conf files.

```
#Enforce upstream server certificate validation at proxy -> #this is not mandated as per CIS buit definitely adds to security. #It requires the
administrator to upload all upstream server certificates to the proxy certificate store #Must-Change Uncomment below lines IF need to enforce
upstream server certificate validation at proxy #proxy_ssl_verify on; #proxy_ssl_trusted_certificate /usr/local/openresty/nginx/ssl/finesse25.crt;
proxy_ssl_trusted_certificate: This file should contain the all upstream certificate enteries concatenated together
```

Caveats for configuring mutual TLS auth:

- Once this feature is enabled on CCBU components client certificate will be asked from LAN clients also during TLS handshake. In case any client/personal certificates are installed on client machine browsers may choose to display a popup to end user asking to choose appropriate certificate for client authentication. Although it does not matter which certificate end user chooses or presses cancel on the popup requests will succeed as client cert auth is not enforced for LAN clients but change in experience will be there. Refer CDET CSCwa26057 for more details.
- Upstream components's webproxy service fails to come up if a proxy host is added to allowed-list which is not resolvable by webproxy service. Make sure reverse proxy hosts added to allowed list are

resolvable from upstream component via DNS lookup.

Clear Cache

The reverse proxy cache can be cleared with the **<NGINX_HOME>/clearCache.sh** command.

Standard Guidelines

This section briefly describes the standard guidelines that need to be followed when you set up Nginx as a proxy server.

These guidelines are derived from the Center for Internet Security. For more details about each guideline, refer to the same.

1. It is always recommended to use the latest stable OpenResty and OpenSSL version.
2. It is advised to install Nginx in a separate disk mount.
3. The Nginx process id must be owned by the root user (or as applicable for chosen OS) and must have permission **644 (rw-------)** or stricter.
4. Nginx must block requests for unknown hosts. Ensure each server block contains the server_name directive explicitly defined. In order to verify, search all server blocks in the **nginx.conf** and **nginx/conf.d** directory and verify that all server blocks contain the server_name.
5. Nginx must listen only on the authorized ports. Search all server blocks in the **nginx.conf** and **nginx/conf.d** directory and check for listen to directives in order to verify only the authorized ports are open for listening.
6. Since Cisco Finesse does not support HTTP, is it recommended to block the proxy server HTTP port as well.
7. The Nginx SSL protocol must be TLS 1.2. Support for legacy SSL protocols must be removed. It also must disable weak SSL ciphers.
8. It is advised for Nginx error and access logs to be sent to the remote syslog server.
9. It is advised to install the **mod_security** module that works as a web application firewall. See the ModSecurity manual for more information. Note that Nginx load has not been verified within the **mod_security** module in place.

Configure the Mapping File

The Reverse proxy deployment of the Finesse desktop requires a mapping file to configure the list of externally visible hostname/port combinations and their mapping to the actual server names and ports that are used by the Finesse, IdS, and CUIC servers. This mapping file which is configured on internal servers is the key configuration that allows the clients connected over the Internet to be redirected to the required hosts and ports that are used on the Internet.

The mapping file has to be deployed on a web server accessible to the component servers and its URI needs to be configured in order for the deployment to work. It is recommended that the mapping file be configured using a dedicated web server available within the network. If such a server is not available, the reverse proxy can be used instead, which will require that the proxy be accessible from within the network and also presents a risk of exposing the information to external clients who can make unauthorized access into the DMZ. The next section details how this can be accomplished.

Refer to the features guide for the exact steps to configure the mapping file URI on all the component servers and for more details on how to create the mapping file data.

Use Reverse Proxy as the Mapping File Server

These steps are required only if the reverse proxy is also used as the proxy mapping file host.

1. Configure the reverse proxy hostname in the domain controller used by the Finesse/CUIC and IdS hosts such that its IP address can be resolved.
2. Upload the generated Nginx signed certificates on both the nodes under tomcat-trust of cmplatform and restart the server.
3. Update the **Must-change** values in **<NGINX_HOME>/html/proxymap.txt**.
4. Reload Nginx configurations with the **nginx -s reload** command.
5. Validate the configuration file is accessible from another network host with the use of the **curl** command.

CentOS 8 Kernel Hardening

If the chosen operating system is CentOS 8, it is recommended that kernel hardening/tuning is done with the use of these sysctl configurations for installations that use a dedicated server for hosting the proxy.

```
## Configurations for kernel hardening - CentOS8. The file path is /etc/sysctl.conf
## Note that the commented configurations denote that CentOS 8's default value matches
## the recommended/tested value, and are not security related configurations.

# Avoid a smurf attack
net.ipv4.icmp_echo_ignore_broadcasts = 1

# Turn on protection for bad icmp error messages
net.ipv4.icmp_ignore_bogus_error_responses = 1

# Turn on syncookies for SYN flood attack protection
net.ipv4.tcp_syncookies = 1

# Turn on and log spoofed, source routed, and redirect packets
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.default.log_martians = 1

# Turn off routing
net.ipv4.ip_forward = 0
net.ipv4.conf.all.forwarding = 0
net.ipv6.conf.all.forwarding = 0

net.ipv4.conf.all.mc_forwarding = 0
net.ipv6.conf.all.mc_forwarding = 0

# Block routed packets
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0
net.ipv6.conf.all.accept_source_route = 0
net.ipv6.conf.default.accept_source_route = 0

# Block ICMP redirects
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv6.conf.all.accept_redirects = 0
net.ipv6.conf.default.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.conf.default.secure_redirects = 0
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0

# Filter routing packets with inward-outward path mismatch(reverse path filtering)
```

```
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1

# Router solicitations & advertisements related.
net.ipv6.conf.default.router_solicitations = 0
net.ipv6.conf.default.accept_ra_rtr_pref = 0
net.ipv6.conf.default.accept_ra_pinfo = 0
net.ipv6.conf.default.accept_ra_defrtr = 0
net.ipv6.conf.default.autoconf = 0
net.ipv6.conf.default.dad_transmits = 0
net.ipv6.conf.default.max_addresses = 1
net.ipv6.conf.all.accept_ra = 0
net.ipv6.conf.default.accept_ra = 0

# Backlog - increased from default 1000 to 5000.
net.core.netdev_max_backlog = 5000

# Setting syn/syn-ack retries to zero, so that they don't stay in the queue.
net.ipv4.tcp_syn_retries = 0
net.ipv4.tcp_synack_retries = 0

# Max tcp listen backlog. Setting it to 511 to match nginx config
net.core.somaxconn = 511

# Reduce the duration of connections held in TIME_WAIT(seconds)
net.ipv4.tcp_fin_timeout = 6

# Maximum resources allotted
# fs.file-max = 2019273
# kernel.pid_max = 4194304
# net.ipv4.ip_local_port_range = 32768 60999

# TCP window size tuning
# net.ipv4.tcp_window_scaling = 1
# net.core.rmem_default = 212992
# net.core.rmem_max = 212992
# net.ipv4.tcp_rmem = 4096 87380 6291456
# net.ipv4.udp_rmem_min = 4096
# net.core.wmem_default = 212992
# net.core.wmem_max = 212992
# net.ipv4.tcp_wmem = 4096 16384 4194304
# net.ipv4.udp_wmem_min = 4096
# vm.lowmem_reserve_ratio = 256 256 32 0 0
# net.ipv4.tcp_mem = 236373 315167 472746

# Randomize virtual address space
kernel.randomize_va_space = 2

# Congestion control
# net.core.default_qdisc = fq_codel
# net.ipv4.tcp_congestion_control = cubic

# Disable SysReq
kernel.sysrq = 0

# Controls the maximum size of a message, in bytes
kernel.msgmnb = 65536

# Controls the default maximum size of a message queue
kernel.msgmax = 65536

# Controls the eagerness of the kernel to swap.
```

```
vm.swappiness = 1
```

A reboot is recommended after you make the recommended changes.

IPtables Hardening

IPtables is an application that allows a system administrator to configure the IPv4 and IPv6 tables, chains, and rules provided by the Linux kernel firewall.

These IPtables rules are configured to secure the proxy application from brute force attacks by restricting the access in the Linux kernel firewall.

The comments in the configuration indicates which service is being rate-limited using the rules.

Note: If administrators use a different port or expand access to multiple servers using the same ports, the appropriate sizing has to be done for these ports accordingly based on these numbers.

```
## Configuration for iptables service
## The file path is /etc/sysconfig/iptables
## Make a note for must-change values to be replaced.
## Restart of the iptable service is required after applying following rules

*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]

# Ensure loopback traffic is configured
-A INPUT -i lo -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A INPUT -s 127.0.0.0/8 -j DROP

# Ensure ping openeded only for the particular source and blocked for rest
# Must-Change: Replace the x.x.x.x with valid ip address
-A INPUT -p ICMP --icmp-type 8 -s x.x.x.x -j ACCEPT

# Ensure outbound and established connections are configured
-A INPUT -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -p tcp -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT

# Block ssh for external interface
# Must-Change: Replace the ens224 with valid ethernet interface
-A INPUT -p tcp -i ens224 --dport 22 -j DROP
# Open inbound ssh(tcp port 22) connections
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT


# Configuration for finesse 8445 port
-A INPUT -p tcp -m tcp --dport 8445 --tcp-flags SYN SYN -m connlimit --connlimit-above 10 --connlimit-m
-A INPUT -p tcp -m tcp --dport 8445 --tcp-flags SYN SYN -m connlimit --connlimit-above 10 --connlimit-m
 -A INPUT -p tcp -m tcp --dport 8445 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 6/sec --hashlimi
-A INPUT -p tcp -m tcp --dport 8445 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst 1 -j LOG -
-A INPUT -p tcp -m tcp --dport 8445 --tcp-flags SYN SYN -j DROP

# Configuration for IdS 8553 port
-A INPUT -p tcp -m tcp --dport 8553 --tcp-flags SYN SYN -m connlimit --connlimit-above 6 --connlimit-ma
```

```
-A INPUT -p tcp -m tcp --dport 8553 --tcp-flags SYN SYN -m connlimit --connlimit-above 6 --connlimit-ma
-A INPUT -p tcp -m tcp --dport 8553 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 2/sec --hashlimit
-A INPUT -p tcp -m tcp --dport 8553 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst 1 -j LOG -
-A INPUT -p tcp -m tcp --dport 8553 --tcp-flags SYN SYN -j DROP

# Configuration for IdP 443 port
-A INPUT -p tcp -m tcp --dport 443 --tcp-flags SYN SYN -m connlimit --connlimit-above 8 --connlimit-mas
-A INPUT -p tcp -m tcp --dport 443 --tcp-flags SYN SYN -m connlimit --connlimit-above 8 --connlimit-mas
-A INPUT -p tcp -m tcp --dport 443 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 4/sec --hashlimit-
-A INPUT -p tcp -m tcp --dport 443 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst 1 -j LOG --
-A INPUT -p tcp -m tcp --dport 443 --tcp-flags SYN SYN -j DROP


# Must-Change: A2A file transfer has not been considered for below IMNP configuration.
# For A2A for support, these configuration must be recalculated to cater different file transfer scenar

# Configuration for IMNP 5280 port
-A INPUT -p tcp -m tcp --dport 5280 --tcp-flags SYN SYN -m connlimit --connlimit-above 30 --connlimit-ma
-A INPUT -p tcp -m tcp --dport 5280 --tcp-flags SYN SYN -m connlimit --connlimit-above 30 --connlimit-ma
-A INPUT -p tcp -m tcp --dport 5280 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 20/sec --hashlimi
-A INPUT -p tcp -m tcp --dport 5280 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst 1 -j LOG -
-A INPUT -p tcp -m tcp --dport 5280 --tcp-flags SYN SYN -j DROP

# Configuration for IMNP 15280 port
-A INPUT -p tcp -m tcp --dport 15280 --tcp-flags SYN SYN -m connlimit --connlimit-above 30 --connlimit-
-A INPUT -p tcp -m tcp --dport 15280 --tcp-flags SYN SYN -m connlimit --connlimit-above 30 --connlimit-
-A INPUT -p tcp -m tcp --dport 15280 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 20/sec --hashlim
-A INPUT -p tcp -m tcp --dport 15280 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst 1 -j LOG
-A INPUT -p tcp -m tcp --dport 15280 --tcp-flags SYN SYN -j DROP


# Configuration for IMNP 25280 port
-A INPUT -p tcp -m tcp --dport 25280 --tcp-flags SYN SYN -m connlimit --connlimit-above 30 --connlimit-
-A INPUT -p tcp -m tcp --dport 25280 --tcp-flags SYN SYN -m connlimit --connlimit-above 30 --connlimit-
-A INPUT -p tcp -m tcp --dport 25280 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 20/sec --hashlim
-A INPUT -p tcp -m tcp --dport 25280 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst 1 -j LOG
-A INPUT -p tcp -m tcp --dport 25280 --tcp-flags SYN SYN -j DROP


# Configuration for CUIC 8444 port
-A INPUT -p tcp -m tcp --dport 8444 --tcp-flags SYN SYN -m connlimit --connlimit-above 6 --connlimit-ma
-A INPUT -p tcp -m tcp --dport 8444 --tcp-flags SYN SYN -m connlimit --connlimit-above 6 --connlimit-ma
-A INPUT -p tcp -m tcp --dport 8444 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 2/sec --hashlimit
-A INPUT -p tcp -m tcp --dport 8444 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst 1 -j LOG -
-A INPUT -p tcp -m tcp --dport 8444 --tcp-flags SYN SYN -j DROP

# Configuration for CUIC 8447 port
-A INPUT -p tcp -m tcp --dport 8447 --tcp-flags SYN SYN -m connlimit --connlimit-above 6 --connlimit-ma
-A INPUT -p tcp -m tcp --dport 8447 --tcp-flags SYN SYN -m connlimit --connlimit-above 6 --connlimit-ma
-A INPUT -p tcp -m tcp --dport 8447 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 2/sec --hashlimit
-A INPUT -p tcp -m tcp --dport 8447 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst 1 -j LOG -
-A INPUT -p tcp -m tcp --dport 8447 --tcp-flags SYN SYN -j DROP

# Configuration for LiveData 12005  port
-A INPUT -p tcp -m tcp --dport 12005 --tcp-flags SYN SYN -m connlimit --connlimit-above 10 --connlimit-
-A INPUT -p tcp -m tcp --dport 12005 --tcp-flags SYN SYN -m connlimit --connlimit-above 10 --connlimit-
-A INPUT -p tcp -m tcp --dport 12005 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 6/sec --hashlimi
-A INPUT -p tcp -m tcp --dport 12005 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst 1 -j LOG
-A INPUT -p tcp -m tcp --dport 12005 --tcp-flags SYN SYN -j DROP

# Configuration for LiveData 12008  port
-A INPUT -p tcp -m tcp --dport 12008 --tcp-flags SYN SYN -m connlimit --connlimit-above 10 --connlimit-
```

```
-A INPUT -p tcp -m tcp --dport 12008 --tcp-flags SYN SYN -m connlimit --connlimit-above 10 --connlimit-
-A INPUT -p tcp -m tcp --dport 12008 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 6/sec --hashlimi
-A INPUT -p tcp -m tcp --dport 12008 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst 1 -j LOG
-A INPUT -p tcp -m tcp --dport 12008 --tcp-flags SYN SYN -j DROP

# Block all other ports
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited

COMMIT
```

These rules could be applied directly by editing the **/etc/sysconfig/iptables** manually or alternatively save the configuration into a file such as **iptables.conf** and execute cat **iptables.conf >>/etc/sysconfig/iptables** to apply the rules.

A restart of the IPtables service is required after you apply the rules. Enter **systemctl restart iptables** in order to restart the IPtables service.

Restrict Client Connections

In addition to the previous IPtables configuration, installations that know the address range for clients who use the proxy are recommended to use this knowledge to secure the proxy access rules. This can provide huge paybacks when it comes to securing the proxy from botnets of malicious networks which are often created in the IP address range of countries which have more lax rules with regards to online security. It is highly recommended, therefore, to restrict the IP address ranges to country /state or ISP-based IP ranges if you are sure of the access patterns.

Block Client Connections

It is also useful to know how to block a specific range of addresses when an attack is identified to be made from an IP address or a range of IP addresses. In such cases, the requests from those IP addresses can be blocked with **iptable** rules.

Block Distinct IP Addresses

In order to block multiple distinct IP addresses, add a line to the **IPTables** configuration file for each IP address.

For example, to block addresses **192.0.2.3** and **192.0.2.4**, enter:

<#root>

```
iptables -A INPUT -s
```

`192.0.2.3`

```
 -j DROP iptables -A INPUT -s
```

`192.0.2.4`

```
 - j DROP.
```

Block a Range of IP Addresses

Block multiple IP addresses in a range and add a single line to the **IPTables** configuration file with the IP range.

For example, to block addresses from 192.0.2.3 to 192.0.2.35, enter:

```
iptables -A INPUT -m iprange --src-range 192.0.2.3-192.0.2.35 -j DROP.
```

Block All IP Addresses in a Subnet

Block all IP addresses in an entire subnet by adding a single line to the **IPTables** configuration file with the use of the classless inter-domain routing notation for the IP address range. For example, to block all class **C addresses**, enter:

```
iptables -A INPUT -s 192.0.0.0/16 -j DROP.
```

SELinux

SELinux is a platform security framework integrated as an enhancement into the Linux OS. The procedure to install and add SELinux policies to run OpenResty as the reverse proxy is provided next.

1. Stop the process with the **openresty -s stop** command.
2. Configure and start /stop nginx server with the **systemctl** command so that during boot up the OpenResty process will start automatically. Enter these commands as root user.
   a. Go to **/usr/lib/systemd/system**.
   b. Open file called **openresty.service**.
   c. Update the content of the file as per **PIDFile** location.

      ```
      [Unit]
      Description=The OpenResty Application Platform
      After=syslog.target network-online.target remote-fs.target nss-lookup.target
      Wants=network-online.target

      [Service]
      Type=forking
      PIDFile=/usr/local/openresty/nginx/logs/nginx.pid
      ExecStartPre=/usr/local/openresty/nginx/sbin/nginx -t
      ExecStart=/usr/local/openresty/nginx/sbin/nginx
      ExecReload=/bin/kill -s HUP $MAINPID
      ExecStop=/bin/kill -s QUIT $MAINPID
      PrivateTmp=true

      [Install]
      WantedBy=multi-user.target
      ```

   d. As root user, enter **sudo systemctl enable openresty.**
   e. Start / Stop the OpenResty service with the **systemctl start openresty / systemctl stop openresty** command and ensure the process is starts / stops as root user.

1. **Install Selinux**

- By default, only some SELinux packages will be installed in CentOs.
- The **policycoreutils-devel** package and its dependencies are required to be installed in order to generate the SELinux policy.
- Enter this command in order to install **policycoreutils-devel**

```
yum install policycoreutils-devel
```

- Ensure that after you install the package, the **sepolicy** command works.

```
usage: sepolicy [-h] [-P POLICY]

                {booleans,communicate,generate,gui,interface,manpage,network,transition}
                ...

SELinux Policy Inspection Tool
```

2. **Create a New Linux User and Map with SElinux User**

a. Enter **semanage login -l** in order to view the mapping between Linux users and SELinux users.

```
[root@loadproxy-cisco-com ~]# semanage login -l

Login Name           SELinux User         MLS/MCS Range        Service

__default__          unconfined_u         s0-s0:c0.c1023       *                *
root                 unconfined_u         s0-s0:c0.c1023       *
```

b. As root, create a new Linux user (**nginx** user) that is mapped to the SELinux **user_u** user.

```
useradd -Z user_u nginxuser
[root@loadproxy-cisco-com ~]# passwd nginxuser
Changing password for user nginxuser.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

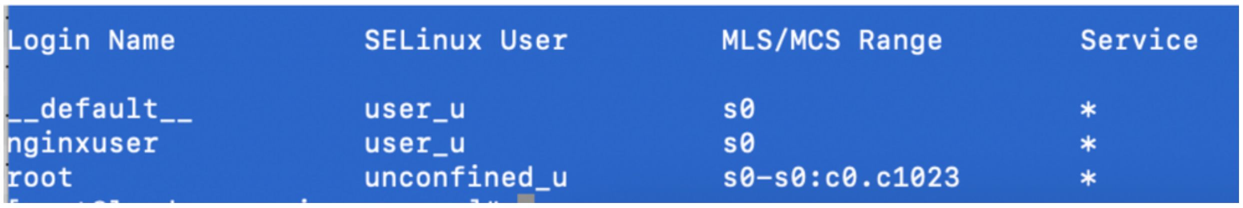
c. In order to view the mapping between **nginxuser** and **user_u**, enter this command as root:

```
[root@loadproxy-cisco-com ~]# semanage login -l

Login Name           SELinux User         MLS/MCS Range        Service

__default__          unconfined_u         s0-s0:c0.c1023       *
nginxuser            user_u               s0                   *
root                 unconfined_u         s0-s0:c0.c1023       *
```

d. SELinux **__default__** login by default mapped to the SELinux **unconfined_u** user. It is required to make **user_u** to be confined by default with this command:

```
semanage login -m -s user_u -r s0 __default__
```

In order to check if the command worked properly, enter **semanage login -l**. It should produce this output:



```
Login Name           SELinux User         MLS/MCS Range        Service

__default__          user_u               s0                   *
nginxuser            user_u               s0                   *
root                 unconfined_u         s0-s0:c0.c1023       *
```

e. Modify nginx.conf and perform change ownership for nginxuser.

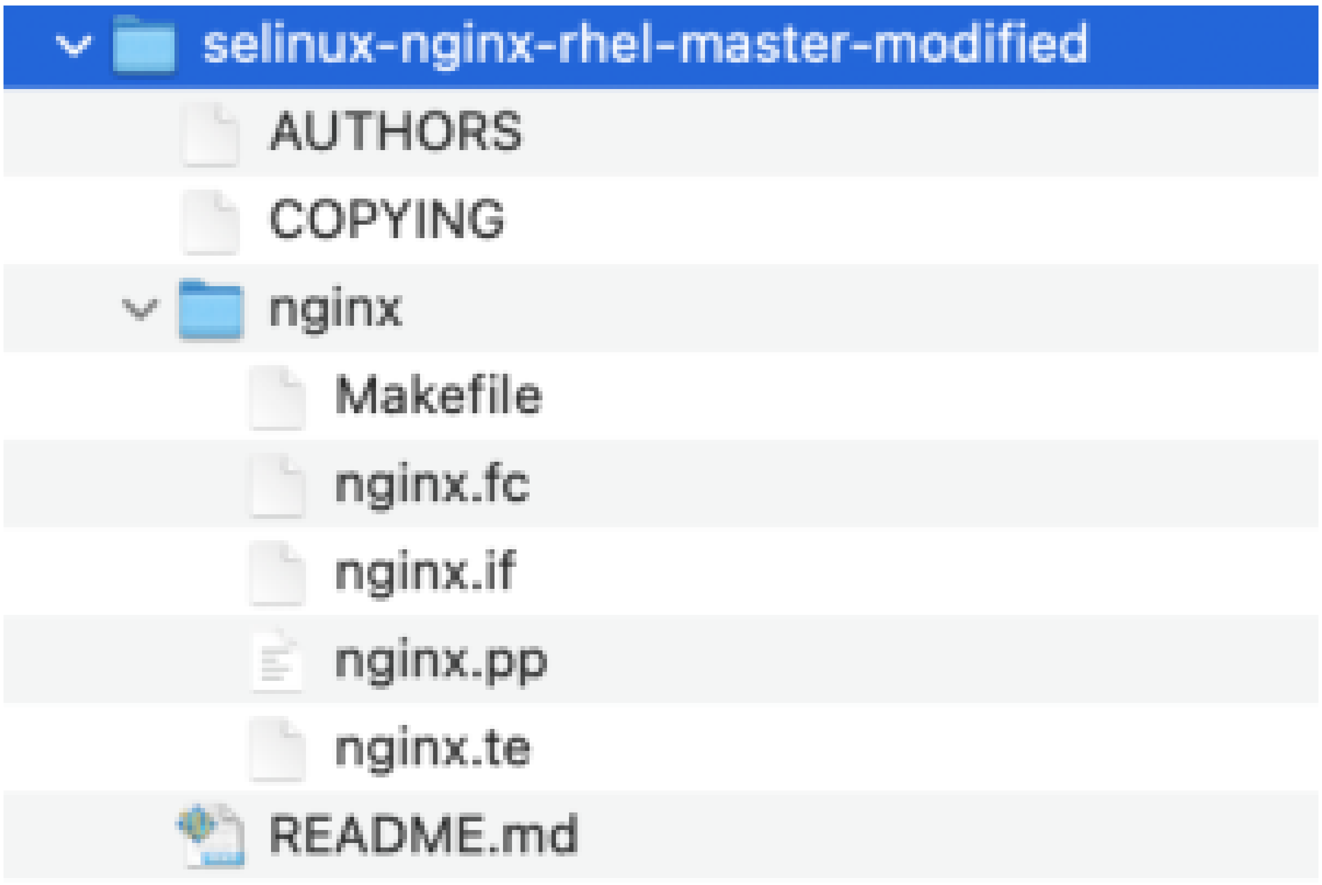
i. Enter **chown -R nginxuser:nginxuser** * in the <**Openresty-install-directory**> directory.

ii. Modify the **nginx.conf** file to include nginxuser as the user for running worker processes.

```
........

user nginxuser nginxuser;

..........
```

Write the SELinux Policy for Nginx

1. Instead of generating a new default custom policy for Nginx with the **sepolicy generate --init /usr/bin/nginx** command, it is prefered to start with an existing policy.
2. The **nginx.fc** file (File Contexts file) and **nginx.te** (Type Enforcement file) files download from the provided URL have been modified to fit the reverse proxy usage.
3. This modified version can be used as a reference since it has been fixed for the particular use case.
4. Download the file **selinux-nginx-rhel-master-modified.tar** from the file software download page.

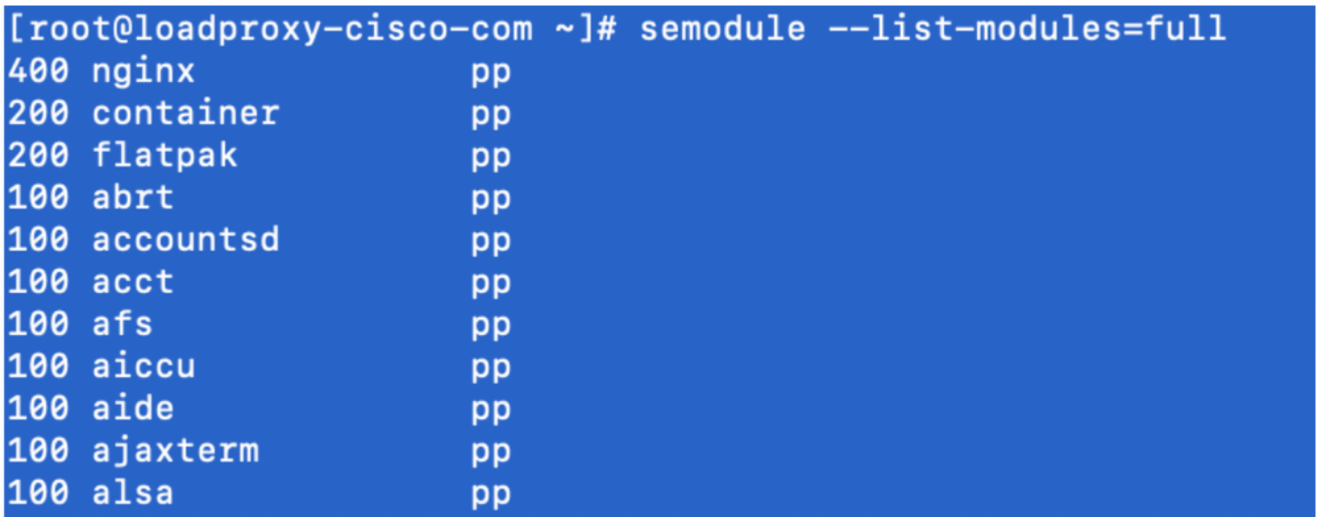


5. Extract the .tar file and navigate to the **nginx** directory within it.
6. Open the **.fc** file and verify the required file paths of nginx installer, cache, and pid file.
7. Compile the configuration with the **make** command.
8. The **nginx.pp** file will be generated.
9. Load the policy with the **semodule** command.

```
semodule -i nginx.pp
```

10. Go to **/root** and create an empty file called **touch /.autorelabel**.
11. Reboot the system.
12. Enter this command in order to verify the policy has been loaded successfully.

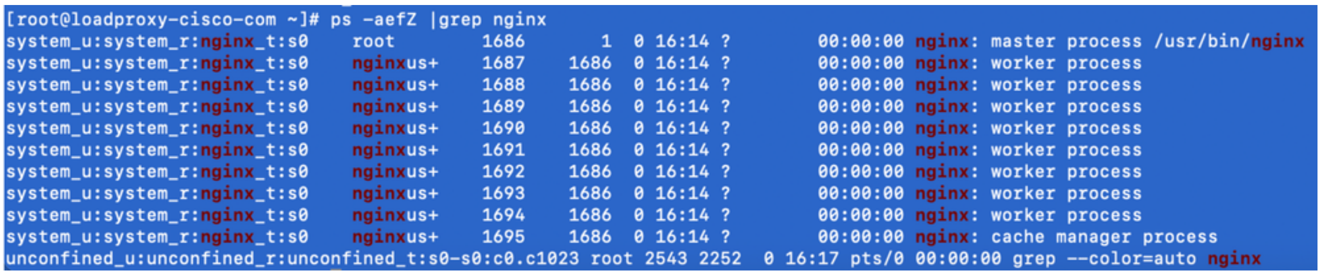
```
semodule --list-modules=full
```



```
[root@loadproxy-cisco-com ~]# semodule --list-modules=full
400 nginx               pp
200 container           pp
200 flatpak             pp
100 abrt                pp
100 accountsd           pp
100 acct                pp
100 afs                 pp
100 aiccu               pp
100 aide                pp
100 ajaxterm            pp
100 alsa                pp
```

13. Nginx should run without any violation. (Violations will be available in **/var/log/messages** and **/var/log/audit/audit.log**).
14. Enter this command in order to check the status of Nginx.

    ```
    ps -aefZ | grep nginx
    ```



```
[root@loadproxy-cisco-com ~]# ps -aefZ |grep nginx
system_u:system_r:nginx_t:s0    root        1686       1  0 16:14 ?        00:00:00 nginx: master process /usr/bin/nginx
system_u:system_r:nginx_t:s0    nginxus+    1687    1686  0 16:14 ?        00:00:00 nginx: worker process
system_u:system_r:nginx_t:s0    nginxus+    1688    1686  0 16:14 ?        00:00:00 nginx: worker process
system_u:system_r:nginx_t:s0    nginxus+    1689    1686  0 16:14 ?        00:00:00 nginx: worker process
system_u:system_r:nginx_t:s0    nginxus+    1690    1686  0 16:14 ?        00:00:00 nginx: worker process
system_u:system_r:nginx_t:s0    nginxus+    1691    1686  0 16:14 ?        00:00:00 nginx: worker process
system_u:system_r:nginx_t:s0    nginxus+    1692    1686  0 16:14 ?        00:00:00 nginx: worker process
system_u:system_r:nginx_t:s0    nginxus+    1693    1686  0 16:14 ?        00:00:00 nginx: worker process
system_u:system_r:nginx_t:s0    nginxus+    1694    1686  0 16:14 ?        00:00:00 nginx: worker process
system_u:system_r:nginx_t:s0    nginxus+    1695    1686  0 16:14 ?        00:00:00 nginx: cache manager process
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 root 2543 2252  0 16:17 pts/0 00:00:00 grep --color=auto nginx
```

15. Now the Finesse agent /supervisor desktop should be accessible.

Verify

Use this section in order to confirm that your configuration works properly.

Finesse

1. Request https://<reverseproxy:port>/finesse/api/SystemInfo. from the DMZ and check if they are reachable.
2. Check **<host>** values in both **<primaryNode>** and **<secondaryNode>** are valid reverse proxy hostnames. It should not be Finesse hostnames.

CUIC and Live Data

1. If Finesse hostnames are seen in the response instead of reverse proxy hostnames, validate proxy-mapping configurations and allowed hosts are properly added in Finesse servers as described in the section "Populate Network Translation Data" of "VPN-Less Access to Finesse Desktop" in the [Finesse 12.6 UCCE Feature Guide.]()
2. If LiveData gadgets load properly in Finesse Desktop, the CUIC and LiveData proxy configurations are proper.
3. In order to validate CUIC and LiveData configuration make HTTP requests to these URLs from the

DMZ and see if they are reachable.
  - https://<reverseproxy:cuic_port>/cuic/rest/about
  - https://<reverseproxy:ldweb_port>/livedata/security
  - https://<reverseproxy:ldsocketio_port>/security

IdS

In order to validate IdS configuration, perform these steps:

1. Log in to the IdSAdmin interface at **https://<ids_LAN_host:ids_port>:8553/idsadmin** from the LAN as the admin interface is not exposed over reverse proxy.
2. Choose **Settings > IdS Trust**.
3. Validate that the proxy cluster publisher node is listed on Download SP metadata page and click **Next**.
4. Validate that the IDP proxy is correctly displayed if configured on Upload IDP metadata page and click **Next**.
5. Initiate test SSO via all proxy cluster nodes from the Test SSO page and validate all are successful. This requires client machine connectivity to reverse proxy nodes.

Performance

The data analysis of top equivalent performance capture, made with the nmon tool, is available from the Finesse Release 12.6(1) ES03 software download page (**load_result.zip**). The data represents the state of the proxy for desktop and supervisor operations, on a sample 2000 UCCE deployment using SSO logins and CUIC LD reports as configured in the default layout for 2000 users for a period of eight hours. It can be used to derive the compute, disk, and network requirements for an installation using Nginx on comparable hardware.

Troubleshoot

SSO

1. **Desktop redirects not going via proxy**
   1. Check that hostnames are configured in correct cases as per actual vm hostnames in various configs like proxymap.txt, server_filter file etc.
   2. Make sure that IdS is added with correct cased hostname in CCE inventory as the same information is pushed to components when registered for SSO from CCE web admin.
2. **SSO logins are not happening**
   1. Make sure IdS-IDP trust is established for proxy host.

SELinux

1. If Nginx is not started by default or the Finesse agent desktop is not accessible, set SELinux to **permissive** mode with this command:

   ```
   setenforce 0
   ```

2. Try to restart the Nginx with the **systemctl restart nginx** command.
3. Violations will be available in **/var/log/messages** and **/var/log/audit/audit.log**.
4. It is required to regenerate .te file with allow rules for addressing those violations by any of these commands:

```
cat /var/log/audit/audit.log | audit2allow -m nginx1 > nginx1.te. # this will create nginx1.te fil
                or
ausearch -c 'nginx' --raw | audit2allow -M  my-nginx  # this will create my-nginx.te file
```

5. Update the original **nginx.te** file present in **selinux-nginx-rhel-master-modified/nginx** directory with newly generated allow rules.
6. Compile the same with the **make** command.
7. The nginx.pp file will be regenerated.
8. Load the policy by semodule command.

```
semodule -i nginx.pp
```

9. Make SELinux to **enforce** mode with this command:

```
setenforce
```

10. Reboot the system.
11. Repeat this procedure until required violations are fixed.