Reverse-Proxy Configuration

- Introduction, on page 1
- Prerequisites, on page 1
- Background Information, on page 2
- Reverse-Proxy Configuration, on page 4
- Verifying Reverse-Proxy Configuration, on page 27
- Brute Force Attack Prevention Configuration, on page 28
- Troubleshoot, on page 30

Introduction

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

Note

- The content in this chapter is provided as a guidance for customers to install and configure reverse-proxy. Cisco does not support requests for reverse-proxy installation and configuration issues. Queries that are related to this subject can be discussed onCisco community forums.
- For ES04 deployments of VPN-less Finesse, see the Cisco Finesse 12.6 ES04 Readme.

Note The OpenResty® Nginx configurations provided as part of Release 12.5(1) SU2 need to be manually edited and applied to match your deployment, along with requiring a manual install of the OpenResty® Nginx.

Prerequisites

Requirements

Cisco recommends that you have knowledge of the following:

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

Components Used

The information in this section is based on the following software and hardware versions:

- Cisco Finesse - 12.6 ES03
- Cisco Unified Intelligence Center - 12.6 ES03
- IdS - 12.6 ES03
- Unified CCE - 11.6 or later
- Packaged CCCE - 12.0 or later

Related Topics

Performance

Background Information

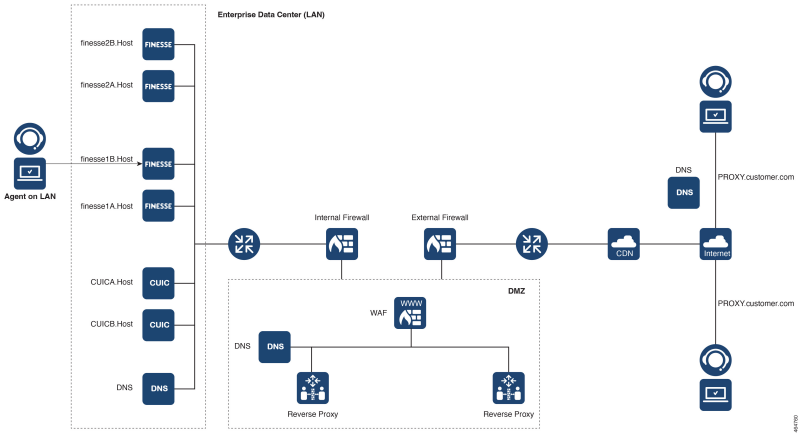
This deployment model is supported for the Unified CCE and Packaged CCE solutions.

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.

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 MRA solution or the Mobile Agent capability of Unified CCE 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 Cisco Unified Intelligence Center nodes through a single HA pair of reverse-proxy nodes.

Concurrent access from agents on the Internet and agents who connect from LAN is supported as shown in the following image:



Note For more information on how to select an appropriate reverse-proxy that supports this deployment, see the section Reverse-Proxy Selection Criteria at *Security Guide for Cisco Unified ICM/Contact Center Enterprise, Release 12.6(1).*

Before you read this section, it is suggested to refer to VPN-less Access to Finesse Desktop. Also, see the *Security Considerations for Mobile Agent Deployments* section in *Security Guide for Cisco Unified ICM/Contact Center Enterprise, Release 12.6(1)*.

Related Topics

Components Used

Upgrade Notes for ES01-Based VPN-less Configurations

- Nginx ES03-based configuration requires Nginx installation with Lua support.
- Certificate Requirements
  - Cisco Finesse, Cisco Unified Intelligence Center, and IdS require the Nginx or OpenResty host certificate to be added to the Tomcat trust store and restart the system. This enables Nginx ES02-based configuration to successfully connect to the component servers.
  - Cisco Finesse, Cisco Unified Intelligence Center, 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-based Nginx configurations.

Note Nginx ES03-based configuration scripts require the corresponding ES03 COP installation in Cisco Finesse, Cisco Unified Intelligence Center, and IdS.

Validating Unauthenticated Static Resources

All valid endpoints that can be accessed without any authentication are actively tracked in the ES04 scripts . Requests to these unauthenticated paths are rejected without sending these requests to the components servers, if an invalid URI is requested.

Brute Force Attack Prevention

Finesse 12.6 ES02 and above authentication scripts actively prevent brute force attacks that can be used to guess the user password. The scripts do 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 number of failed requests, time interval, and blocking duration are configurable.

For more information, see the Brute Force Attack Prevention Details section.

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.

Reverse-Proxy Configuration

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 in *Background Information*.

1. Install the OS of your choice with the required hardware specification. Kernel and IPv4 parameter tweaks might differ depending on the OS selected. Users are advised to reverify these aspects if the chosen OS version is different from CentOS 8.0.
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 of the following Nginx versions 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 must be compiled along with OpenResty-based Lua modules)
- 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.

Install OpenResty

Procedure

	Command or Action	Purpose
Step 1	Install OpenResty. See OpenResty Linux Packages.	As part of the OpenResty installation, Nginx will be installed in this location. Add the OpenResty path to the ***PATH*** variable by adding the following line in the ***~/.bashrc*** file. `export PATH=/usr/local/openresty/bin:$PATH`
Step 2	Start or stop OpenResty Nginx	• To start OpenResty Nginx, enter **`openresty`**. • To stop OpenResty Nginx, enter **`openresty -s stop`**.

Configure OpenResty Nginx

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

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

1. Download and extract the 12.6-ES04-reverse-proxy-config.zip file that contains the reverse-proxy configuration for Nginx. This file is available on the Finesse Release 12.6(1)ES04 software download page.
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 OpenResty 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 OpenResty Nginx defaults are not used.

6. OpenResty 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 the 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 that the cache directories configured for Cisco Unified Intelligence Center 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 Agent deployment and has to be expanded appropriately for a larger deployment.

Configure the OpenResty 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 server. The same steps should be followed for the secondary Finesse and Cisco Unified Intelligence Center 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 `mount -av` command.
3. Use the `df -h` command to validate if the file system has mounted the new mount points.
4. Change the proxy_cache_path locations in the Finesse and Cisco Unified Intelligence Center 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 server and Cisco Unified Intelligence Center server.

Note Ensure that the sum of all the **tmpfs** drive sizings created in all the previous steps are added to the final memory sizing for the deployment. This is because these drives are memory blocks that are configured to look like disks to the application and they consume memory.

Configure Log Rotation

Nginx reverse-proxy produces many logs. Configure log rotation to ensure optimum use of disk space, else the logs fill up the disk. The steps to configure log rotation is as follows:

1. Copy the configuration file from `/usr/local/openresty/nginx/logrotate/saproxy` to `/etc/logrotate.d/reverseproxy`.
2. Ensure that there's a file in the `/etc/cron.daily/logrotate`. This does the log rotation daily based on the configuration in the `/etc/logrotate.d/reverseproxy`.

Note If log rotation has to be done more frequently, such as every hour, copy the configuration file from `/etc/cron.daily/logrotate` to `/etc/cron.hourly/logrotate`.

3. Log files under `/usr/local/openresty/nginx/logs/` are rotated based on the configuration in `/etc/logrotate.d/reverseproxy`. That is, rotate the log when the file size exceeds 100 MB and keep no more than 20 most recently rotated files.

The status of the log rotation is available in the `/var/lib/logrotate/logrotate.status` log file.

Use Self-Signed Certificates—Test Deployments

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

1. Generate OpenResty Nginx certificates for SSL folder content. Before you generate certificates, you must create a folder called **ssl** under `/usr/local/openresty/nginx`. Generate two certificates (one for `<reverseproxy_primary_fqdn>` and another for <`reverseproxy_secondary_fqdn`>) with the help of the following commands:

   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 hostname 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 hostname 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`, because these are already configured in Finesse Nginx configuration files.

2. Change the permission of the private key **400 (r--------)**.

3. Configure the firewall and iptables on the reverse-proxy to enable the firewall to communicate with the ports that are configured to listen to the OpenResty Nginx server.

4. Add the IP address and hostname of all the configured servers in the `/etc/hosts` file of the reverse-proxy server.

Note The provided configuration is for a sample 2000 Agent deployment and must be expanded appropriately for larger deployments.

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).

   To generate the CSR and private key, enter `openssl req -new -newkey rsa:4096 -keyout nginx.key -out nginx.csr` after you log in to 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 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. This is 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 in 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 or add the following SSL configurations related to the certificate in the OpenResty Nginx configurations in the following 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 OpenResty Nginx.

Create Custom Diffie-Hellman Parameter

1. Create a custom Diffie-Hellman parameter by using the following commands:

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

2. Modify the server configuration to use the new parameters in the file `/usr/local/openresty/nginx/conf/conf.d/ssl/ssl.conf` by using the following command:

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

Enable OCSP Stapling

Note In order to enable the Online Certificate Status Protocol (OCSP) stapling, the server should be using a CA-signed certificate and the server should have access to the CA which signed the certificate.

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

- `ssl_stapling on;`
- `ssl_stapling_verify on;`

Modify the Common OpenResty Nginx Configuration

The default OpenResty Nginx configuration file `(/usr/local/openresty/nginx/conf/nginx.conf)` has to be modified to contain the following entries to enforce security and enhance performance. The following content should be used to modify the default configuration file (`nginx.conf`) which is created during the OpenResty Nginx installation:

```
# Increasing number of worker processes will not increase the processing the request. The
number of worker process will be same as number of cores
# in system CPU. OpenResty Nginx provides "auto" option to automate this, which will spawn
 one worker for each CPU core.
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 our proxy set up the value is 0
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_connections

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 maximum number of open files for the user (ulimit -Hn)
    # The appropriate setting depends on the size of the server and the nature of the
traffic, and can be discovered through testing.
    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_manager")
        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/api/Users")
           UnauthenticatedDesktopResourcesManager.getDesktopResources("saproxy.cisco.com",
 "https://saproxy.cisco.com:8445/desktop/api/urls?type=desktop")
          UnauthenticatedResourcesManager.getThirdPartyGadgetResources("saproxy.cisco.com",
 "https://saproxy.cisco.com:8445/desktop/api/urls?type=3rdParty")
        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 for Finesse requests on port 8445. At a time, only one port can be enabled from a reverse-proxy to support Finesse requests. If port 443 needs to be supported, edit the `<nginx-install-directory>conf/conf.d/finesse.conf` file to enable listening on 443 and disable listening on 8445.

Configure Mutual TLS Authentication Between Reverse-Proxy and Components

Client SSL certificate authentication for connections from reverse-proxy hosts can be enabled on Cisco Unified Intelligence Center, Finesse, IdS, and LiveData by using the new CVOS CLI option `utils system reverse-proxy client-auth enable/disable/status`.

By default this is disabled and has to be enabled by the administrator by executing the CLI on each upstream server independently. After 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 by using the CLI `utils system reverse-proxy allowed-hosts add <proxy-host>`.

The following is the configuration block for the same in proxy config files named **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;
```

The SSL certificate used for outbound traffic (proxy to upstream) can be the same as the SSL certificate that is configured for inbound traffic (SSL connector for component server blocks). If self-signed certificate is used as **proxy_ssl_certificate**, it has to be uploaded to the tomcat trust store of the upstream components (Finesse/IdS/Cisco Unified Intelligence Center/Livedata) 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, the following configuration has to be uncommented in the **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
```

mutual TLS (mTLS) is a standard security requirement for connections established from DMZ into the data center. For more information, see Nginx CIS behcmarks-https://www.cisecurity.org/benchmark/nginx

mTLS requires that both the server and client be pre-configured with mutual information about each other, as well as that the mutual certificates be properly verified. Hence the term Mutual TLS. A properly configured proxy server will be able to circumvent TCP rate limits and provide the client IP to the server for logging

purposes. As a result, it is critical that the proxy identity be verified before connecting as a reverse-proxy. For security reasons, it is therefore recommended that this feature be used and turned on.

This requires the upstream component certificates to be made available to the proxy and vice-versa. Reverse-proxy by default establishes verified TLS connections to the upstream server and it is the proxy verification at the client which is optional. Therefore this needs to be enabled at the upstream client server.

Enabling mutual TLS

The mutual TLS needs to be enabled at the upstream component servers using the provided CLI.

Use the **utils system reverse-proxy client-auth enable** CLI to enable proxy certificate verification at the upstream component server.

After running the CLI, upload the proxy SSL certificate corresponding to the reverse-proxy hostname used to connect to the same server. This can be used to verify TLS connections when the reverse-proxy attempts to establish an upstream connection.

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 must be followed when you set up OpenResty Nginx as a proxy server. The guidelines for the OpenResty Nginx server software is derived from the Center for Internet Security.

1. Use the latest stable versions of OpenResty and OpenSSL version.
2. Install OpenResty Nginx in a separate disk mount.
3. The OpenResty Nginx process id must be owned by the root user (or as applicable for the chosen OS) and must have permission **644 (rw-------)** or stricter.
4. OpenResty Nginx must block requests for unknown hosts. Ensure that each server block contains the server_name directive explicitly defined. To verify, search all server blocks in the `nginx.conf` and `nginx/conf.d` files and verify that all server blocks contain the server_name.
5. OpenResty Nginx must listen only on the authorized ports. Search all server blocks in the `nginx.conf` and `nginx/conf.d` files and check for the `listen` directives to verify that only the authorized ports are open for requests.
6. Block the proxy server HTTP port, because Cisco Finesse does not support HTTP.
7. The OpenResty Nginx SSL protocol must be TLS 1.2. Remove support for legacy SSL protocols. Disable weak SSL ciphers.
8. Send the OpenResty Nginx error and access logs to the remote syslog server.
9. Install the **mod_security** module that works as a web application firewall. See the ModSecurity manual for more information. Note that OpenResty Nginx load has not been verified within the **mod_security** module in place.

Configure the Mapping File

Refer to Host Mapping File for Network Translation.

Related Topics

Host Mapping File for Network Translation

Use Reverse-Proxy as the Mapping File Server

Note This appendix has the configuration details, for more information about the pre-requisites, refer to 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, Cisco Unified Intelligence Center and IdS hosts such that its IP address can be resolved.
2. Upload the generated OpenResty® 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 OpenResty® Nginx configurations with the `nginx -s reload` command.
5. Use the `curl` command to validate if the configuration file is accessible from another network host.

CentOS 8 Kernel Hardening

If the operating system is Cent OS 8 and the installations use a dedicated server for hosting the proxy, harden the kernel by using these `sysctl` configurations:

```
## 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
```

Reboot 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.

The 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 indicate which service is being rate-limited by using the rules.

Note If administrators use a different port or expand access to multiple servers using the same ports, they must do appropriate sizing for these ports accordingly.

A sample IPtable is as follows:

```
## 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 ccp 8445 port
-A INPUT -p tcp -m tcp --dport 8445 --tcp-flags SYN SYN -m connlimit --connlimit-above 10
--connlimit-mask 32 --connlimit-saddr -m limit --limit 1/min --limit-burst 1 -j LOG
--log-prefix " Connections to 8445 exceeded connlimit "
-A INPUT -p tcp -m tcp --dport 8445 --tcp-flags SYN SYN -m connlimit --connlimit-above 10
--connlimit-mask 32 --connlimit-saddr -j DROP
 -A INPUT -p tcp -m tcp --dport 8445 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 6/sec
 --hashlimit-burst 8 --hashlimit-mode srcip,dstport --hashlimit-name TCP_8445_DOS -j ACCEPT
-A INPUT -p tcp -m tcp --dport 8445 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst
 1 -j LOG --log-prefix " Exceeded 8445 hashlimit "
-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-mask 32 --connlimit-saddr -m limit --limit 1/min --limit-burst 1 -j LOG
--log-prefix " IdS connection limit exceeded"
-A INPUT -p tcp -m tcp --dport 8553 --tcp-flags SYN SYN -m connlimit --connlimit-above 6
--connlimit-mask 32 --connlimit-saddr -j DROP
-A INPUT -p tcp -m tcp --dport 8553 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 2/sec
 --hashlimit-burst 4 --hashlimit-mode srcip,dstport --hashlimit-name TCP_8553_DOS -j ACCEPT
-A INPUT -p tcp -m tcp --dport 8553 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst
 1 -j LOG --log-prefix " Exceeded 8553 hashlimit "
-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-mask 32 --connlimit-saddr -m limit --limit 1/min --limit-burst 1 -j LOG
--log-prefix " IdP connection limit exceeded"
-A INPUT -p tcp -m tcp --dport 443 --tcp-flags SYN SYN -m connlimit --connlimit-above 8
--connlimit-mask 32 --connlimit-saddr -j DROP
-A INPUT -p tcp -m tcp --dport 443 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 4/sec
 --hashlimit-burst 6 --hashlimit-mode srcip,dstport --hashlimit-name TCP_443_DOS -j ACCEPT
-A INPUT -p tcp -m tcp --dport 443 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst
 1 -j LOG --log-prefix " Exceeded 443 hashlimit "
-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 scenarios.

# Configuration for IMNP 5280 port
-A INPUT -p tcp -m tcp --dport 5280 --tcp-flags SYN SYN -m connlimit --connlimit-above 30
--connlimit-mask 32 --connlimit-saddr -m limit --limit 1/min --limit-burst 1 -j LOG
--log-prefix " IMNP connection limit exceeded"
-A INPUT -p tcp -m tcp --dport 5280 --tcp-flags SYN SYN -m connlimit --connlimit-above 30
--connlimit-mask 32 --connlimit-saddr -j DROP
-A INPUT -p tcp -m tcp --dport 5280 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 20/sec
 --hashlimit-burst 25 --hashlimit-mode srcip,dstport --hashlimit-name TCP_5280_DOS -j ACCEPT
-A INPUT -p tcp -m tcp --dport 5280 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst
 1 -j LOG --log-prefix " Exceeded 5280 hashlimit "
-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-mask 32 --connlimit-saddr -m limit --limit 1/min --limit-burst 1 -j LOG
--log-prefix " IMNP connection limit exceeded"
-A INPUT -p tcp -m tcp --dport 15280 --tcp-flags SYN SYN -m connlimit --connlimit-above 30
 --connlimit-mask 32 --connlimit-saddr -j DROP
-A INPUT -p tcp -m tcp --dport 15280 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto
20/sec --hashlimit-burst 25 --hashlimit-mode srcip,dstport --hashlimit-name TCP_15280_DOS
-j ACCEPT
```

```
-A INPUT -p tcp -m tcp --dport 15280 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst
 1 -j LOG --log-prefix " Exceeded 15280 hashlimit "
-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-mask 32 --connlimit-saddr -m limit --limit 1/min --limit-burst 1 -j LOG
--log-prefix " IMNP connection limit exceeded"
-A INPUT -p tcp -m tcp --dport 25280 --tcp-flags SYN SYN -m connlimit --connlimit-above 30
 --connlimit-mask 32 --connlimit-saddr -j DROP
-A INPUT -p tcp -m tcp --dport 25280 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto
20/sec --hashlimit-burst 25 --hashlimit-mode srcip,dstport --hashlimit-name TCP_25280_DOS
-j ACCEPT
-A INPUT -p tcp -m tcp --dport 25280 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst
 1 -j LOG --log-prefix " Exceeded 25280 hashlimit "
-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-mask 32 --connlimit-saddr -m limit --limit 1/min --limit-burst 1 -j LOG
--log-prefix " CUIC connection limit exceeded"
-A INPUT -p tcp -m tcp --dport 8444 --tcp-flags SYN SYN -m connlimit --connlimit-above 6
--connlimit-mask 32 --connlimit-saddr -j DROP
-A INPUT -p tcp -m tcp --dport 8444 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 2/sec
 --hashlimit-burst 4 --hashlimit-mode srcip,dstport --hashlimit-name TCP_8444_DOS -j ACCEPT
-A INPUT -p tcp -m tcp --dport 8444 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst
 1 -j LOG --log-prefix " Exceeded 8444 hashlimit "
-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-mask 32 --connlimit-saddr -m limit --limit 1/min --limit-burst 1 -j LOG
--log-prefix " CUIC connection limit exceeded"
-A INPUT -p tcp -m tcp --dport 8447 --tcp-flags SYN SYN -m connlimit --connlimit-above 6
--connlimit-mask 32 --connlimit-saddr -j DROP
-A INPUT -p tcp -m tcp --dport 8447 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 2/sec
 --hashlimit-burst 4 --hashlimit-mode srcip,dstport --hashlimit-name  TCP_8447_DOS -j ACCEPT
-A INPUT -p tcp -m tcp --dport 8447 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst
 1 -j LOG --log-prefix " Exceeded 8447 hashlimit "
-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-mask 32 --connlimit-saddr -m limit --limit 1/min --limit-burst 1 -j LOG
--log-prefix " LD connection limit exceeded"
-A INPUT -p tcp -m tcp --dport 12005 --tcp-flags SYN SYN -m connlimit --connlimit-above 10
 --connlimit-mask 32 --connlimit-saddr -j DROP
-A INPUT -p tcp -m tcp --dport 12005 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 6/sec
 --hashlimit-burst 8 --hashlimit-mode srcip,dstport --hashlimit-name TCP_12005_DOS -j ACCEPT
-A INPUT -p tcp -m tcp --dport 12005 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst
 1 -j LOG --log-prefix " Exceeded 12005 hashlimit "
-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-mask 32 --connlimit-saddr -m limit --limit 1/min --limit-burst 1 -j LOG
--log-prefix " LD connection limit exceeded"
-A INPUT -p tcp -m tcp --dport 12008 --tcp-flags SYN SYN -m connlimit --connlimit-above 10
 --connlimit-mask 32 --connlimit-saddr -j DROP
-A INPUT -p tcp -m tcp --dport 12008 --tcp-flags SYN SYN -m hashlimit --hashlimit-upto 6/sec
 --hashlimit-burst 8 --hashlimit-mode srcip,dstport --hashlimit-name  TCP_12008_DOS -j
ACCEPT
```

```
-A INPUT -p tcp -m tcp --dport 12008 --tcp-flags SYN SYN -m limit --limit 1/min --limit-burst
 1 -j LOG --log-prefix " Exceeded 12008 hashlimit "
-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
```

Note The rules that are provided block the DNS resolution at the proxy. So, all the hostnames of the components that are configured in the proxy must be explicitly added to the host resolution file `/etc/hosts`.

Interface level rules must be added to restrict access to only users accessing via LAN and to block public access to port 10000, which is used for accessing the proxy map file. For example,

```
-A INPUT -p tcp -m tcp -i <PRIVATE_INTERFACE> --dport 10000 --tcp-flags SYN SYN -m hashlimit
--hashlimit-upto 35/sec --hashlimit-burst 2000 --hashlimit-mode srcip,dstport --hashlimit-name
TCP_10000_DOS -j ACCEPT -A INPUT -p tcp -m tcp -i <PRIVATE_INTERFACE> --dport 10000
--tcp-flags SYN SYN -m limit --limit 1/min --limit-burst 1 -j LOG --log-prefix " Exceeded
hashlimit " -A INPUT -p tcp -m tcp -i <PRIVATE_INTERFACE> --dport 10000 --tcp-flags SYN SYN
-j DROP
```

These rules could be applied directly by editing the `/etc/sysconfig/iptables` file manually. Alternatively, save the configuration into a file such as `iptables.conf` and run cat `iptables.conf >>/etc/sysconfig/iptables` to apply the rules.

Restart the IPtables service after you apply the rules. To restart the IPtables service, enter `systemctl restart iptables`.

Restrict Client Connections

In addition to the previous IPtables configuration, installations that know the address range for clients who use the proxy must use this knowledge to secure the proxy access rules. This helps to secure the proxy from malicious botnets which are often created in the IP address range of countries that have more lax rules with regards to online security. Restrict the IP address ranges to country-based, state-based, or ISP-based IP ranges if you are sure of the access patterns.

Block Client Connections

Block the 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

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

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

```
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 address range.

For example, to block the 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 by using 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

Security-Enhanced Linux (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 or stop OpenResty 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 the 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 or stop the OpenResty service with the `systemctl start openresty / systemctl stop openresty` command and ensure that the process starts or 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 must be installed in order to generate the SELinux policy.
- Enter the following command 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` 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.
```

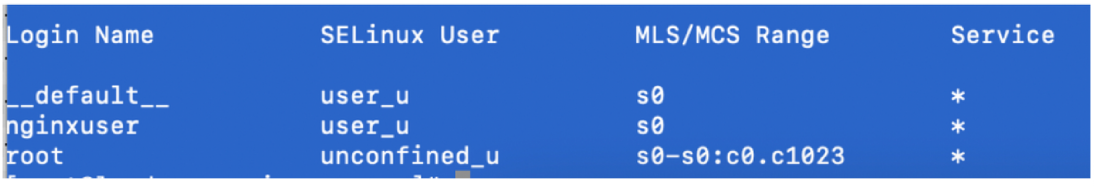
   c. In order to view the mapping between **nginxuser** and **user_u**, enter this command as root:

```
[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 is by default mapped to the SELinux **unconfined_u** user. By default, it is required to confine **user_u** by using the following 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:



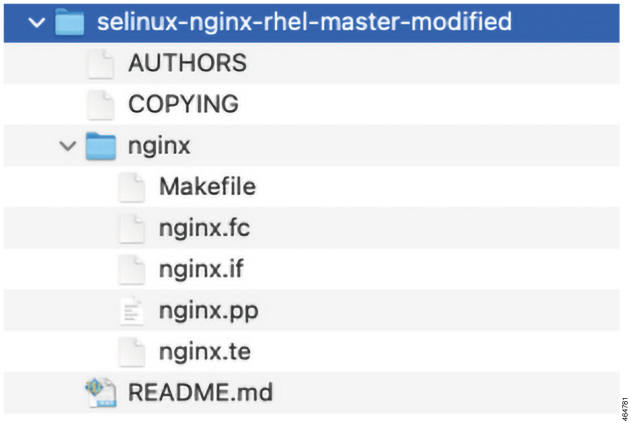
   e. Modify nginx.conf and perform change ownership for nginxuser.

1. Enter `chown -R nginxuser:nginxuser *` in the `<Openresty-install-directory>` directory.
2. Modify the **nginx.conf** file to include nginxuser as the user for running worker processes.

```
........

user nginxuser nginxuser;

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

3. **Write the SELinux Policy for OpenResty Nginx**
   a. Instead of generating a new default custom policy for OpenResty Nginx with the `sepolicy generate --init /usr/bin/nginx` command, start with an existing policy.

   The **nginx.fc** file (File Contexts file) and **nginx.te** (Type Enforcement file) files, that are downloaded from the following location, are modified for reverse-proxy usage:



   This modified version can be used as a reference because it is updated for a particular use case.

   b. Download the **selinux-nginx-rhel-master-modified.tar** file from the Software Download .

   c. Extract the **.tar** file and navigate to the **nginx** directory within it.

   d. Open the **.fc** file and verify the required file paths of **Nginx installer**, **cache**, and **pid** files.

   e. Compile the configuration with the `make` command.

   f. The **nginx.pp** file is generated.

   g. Load the policy with the `semodule` command.

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

   h. Go to **/root** and create an empty file called `touch /.autorelabel`.

   i. Reboot the system.

   j. Enter the following command to verify that the policy is 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
```

464782

k. OpenResty Nginx should run without any violation. (Violation logs will be available in **/var/log/messages** and **/var/log/audit/audit.log**).

l. Enter the following command to check the status of OpenResty 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
```

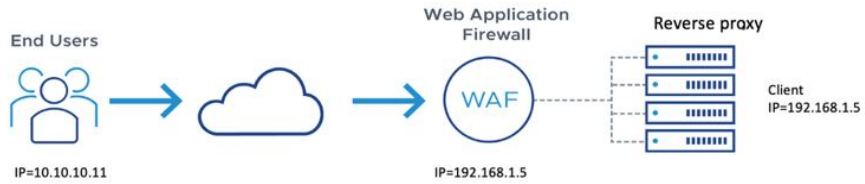
464784

m. Now the Finesse agent desktop or supervisor desktop should be accessible.

Load Balancer, WAF, and Proxy support for reverse-proxy deployments

The reverse-proxy configurations have security features that are dependent on the information of the actual client IP. Requesting rate limits, logging of client activity, blocking the users due to multiple wrong credentials require the configuration to track the client IP to appropriately rate-limit, block, or log the actual users' access.

No specific configurations are required for deployments which directly terminate the agent connections on the reverse-proxy. The reverse-proxy has the information of the client IP due to the connections directly reaching the reverse-proxy. However, when other network devices are used to terminate the client connections, before forwarding them as fresh requests to the reverse-proxy, the client IPs are no longer visible to the reverse-proxy.



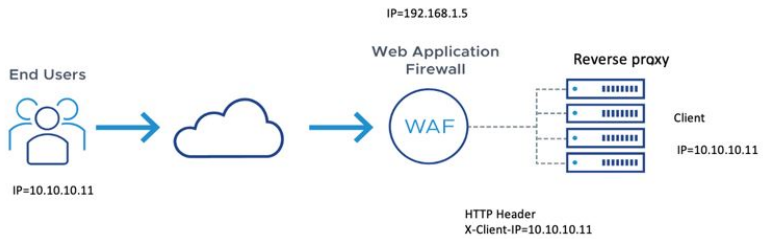
This happens when there are Load Balancers, Web Application Firewall (WAF), or other proxies deployed in front of a reverse-proxy. The CDN deployments work as an intermediary reverse-proxy/WAF and fall into the same deployment category.

Such deployments **MUST** add certain reverse-proxy configurations to enable the reverse-proxy to identify the actual client IP. The configurations that are required for such deployments are as follows:

1. The public IPs and private IPs of the devices which will forward the requests to the reverse-proxy must be to be added in **nginx.conf** http block and **maps.conf** geo block as mentioned in the **##Must-change** comment.
2. The new requests originating from the intermediary devices, **MUST** populate HTTP request header fields with the end-client IP to communicate the same to the reverse-proxy.

   The name of the request header field is configured in **nginx.conf** file using `real_ip_header` directive. For example: `real_ip_header X-REAL-IP;`

Note All CDN deployments provide a mechanism to extract the client IP as a HTTP header containing a single-client IP as part of the request payload. A custom header is often recommended to avoid conflict with the standard `X-FORWARDED-FOR` header. The VPN-less reverse-proxy deployments are also recommended to provide the client IP using a custom header for similar reasons.



3. For security purpose, the devices which are front ending the reverse-proxy **MUST** replace `X-FORWARDED-FOR` and `X-REAL-IP` headers provided by the client with the actual client IP or drop them if the deployment does not need these headers.
4. If the deployment is using a custom HTTP header for communicating to the client IP, the particular field **MUST** be replaced with the client IP before forwarding them upstream to the reverse-proxy.
5. Verify the configuration by transmitting a high rate of requests to a Finesse API such as `SystemInfo/DesktopConfig` from an external client. Verify through the Load Balancer or WAF to ensure that the client is blocked while the Load Balancer or intermediate devices are not blocked or rate limited. Ensure that the configurations are working as expected before going live.

   Refer to the Logging section for instructions on how to check whether a client is blocked or rate limited.
6. Deployments that employ WAF or other security devices must ensure that the desktop API traffic patterns are compatible with them before going live with the deployment. Certain WAF rules can be too restrictive and may need some modifications before they can be deployed.

Note The reverse-proxy configurations provided have no protection against layer-3 attacks such as IP address spoofing or flooding. The proxy provides only rate limiting, brute force attack detection, and restricting of requests to the allowed destinations. The operating system IP configurations are hardened to a certain level but there are no further protections that are available. It is assumed that the relevant operating system hardening and traffic protection devices are employed to secure the deployment Cisco Contact Center.

For more details refer to the *Security Guidelines for Reverse-Proxy Deployment* section in the *Security Guide for Cisco Unified ICM/Contact Center Enterprise, Release 12.6(1)* guide.

Load Balancers and other devices which does not have the HTTP header support can skip second and third points that are mentioned above. However, this causes a sub-optimal deployment which will be functional but loses certain features such as client IP logging for debugging purposes and blocking users attempting to brute force guess passwords.

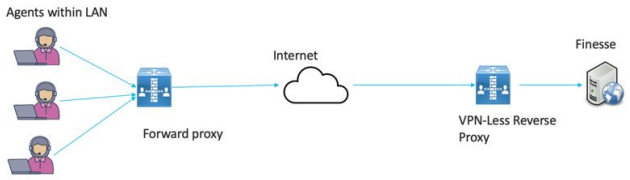
The websocket authentications will also be not effective at the reverse-proxy, which will not cause any loss in functionality but will allow all websocket request to reach the upstream component before authentication can be enforced.

Related Topics

Logging
Troubleshooting

Access VPN-Less proxy through Forward proxy and NAT

The VPN-less configuration assumes that the proxy is accessed by clients/agents from the internet, who have separate individual IP's which can be used for enforcing security features. However, not all deployments dedicatedly use agents from the internet with their own unique IP addresses. Most deployments will have agents accessing the reverse-proxy deployments both from the internet as well as from LAN using the same reverse-proxy access URI.



So, if you have a deployment which uses agents behind a proxy or a NAT that looks like what is shown above, certain configuration changes have to be made to ensure that the end-user IP's are correctly communicated to the reverse-proxy. The steps to configure are as follows:

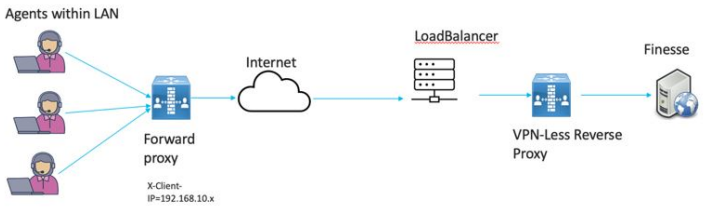
1. The Forward proxy (device A in the diagram above) has to be well-known in advance.
2. The Forward proxy device has to transmit the agent IP’s in a predefined header. For example, `X-REAL-IP` as shown above.
3. If there are other intermediary devices such as a Load Balancer or WAF at the network where Finesse is deployed, before the requests reach the reverse-proxy, these devices must be able to allow the Forward proxy by its IP address and then transmit the HTTP header without any changes.

Note It is crucial that the Forward proxy IP address is identified and only requests from this IP should be allowed to have the `X-REAL-IP` transmitted.

4. The **nginx.conf** http block and **maps.conf** geo block files should be updated with the list of Forward proxies' private and public IPs as mentioned in the **##Must-change** comments.

Note Deployments that do not have the HTTP header support can skip the steps 2 to 4. However, this causes a sub-optimal deployment which will be functional but loses certain security features listed above, which are dependent on client IP knowledge and these deployments are therefore not suggested.

Ensure that the deployment cannot support multiple HTTP header names to transmit the client IP corresponding to different Forward proxies that the network is interacting with.



Deployments such as these, should transmit or detect the final client IP of the users who are connecting from behind the **Forward proxy A** and this would be an agreement between Load Balancer and the Forward Proxy.

The Load Balancer or the final intermediary devices that forward requests to the VPN-less reverse-proxy should transmit the required headers and will need configuration as described in the section above. The Forward proxy information is not required to be added to the VPN-less configuration, if the intermediary device is able to identify the correct client IPs and transmit them to the reverse-proxy using the steps mentioned above.

However, if the actual client IP resolution is not setup between the Forward proxy and the Load Balancer, the reverse-proxy considers the IP of the Forward proxy as the actual client IP.

In this case, to avoid rate limiting to block the Forward proxy, its private and public IP addresses must be configured in **nginx.conf** http block and **maps.conf** geo block files. Both the files must be updated with the list of Forward-proxies' IP as mentioned in the **##Must-change** comments, so that the proxy is not blocked or rate limited. This would be a sub-optimal deployment and transmitting the actual client IP is recommended for a more effective deployment.

Verifying Reverse-Proxy Configuration

Finesse

Procedure

Step 1 From the DMZ, open *https://<reverseproxy:port>/finesse/api/SystemInfo* and check if it's reachable.

Step 2 Check if the **<host>** values in both **<primaryNode>** and **<secondaryNode>** are valid in reverse-proxy hostnames. It shouldn't be Finesse hostnames.

Note
- If CORS status is "enabled", you must explicitly add the reverse-proxy domain name to the list of CORS trusted domain names.
- Reverse-proxy supports a maximum of 8000 folders (including subdirectories) in the `finesse/3rdpartygadget` folder.

Cisco Unified Intelligence Center and LiveData

Procedure

Step 1 If the Finesse hostnames are seen in the response instead of reverse-proxy hostnames, validate the proxy-mapping configurations and check if the allowed hosts are properly added in Finesse servers as described in the section *Populate Network Translation Data*.

Step 2 If LiveData gadgets load properly in Finesse Desktop, then CUIC and LiveData proxy configurations are proper.

Step 3 To validate the Cisco Unified Intelligence Center and LiveData configurations, make the HTTP requests from the DMZ to the following URLs and check if they are reachable:

- https://<reverseproxy:cuic_port>/cuic/rest/about
- https://<reverseproxy:ldweb_port>/livedata/security
- https://<reverseproxy:ldsocketio_port>/security

Cisco Identity Service

To validate Cisco IdS configuration, perform the following steps:

Procedure

Step 1 Log in to the IdSAdmin interface at **https://<ids_LAN_host:ids_port>:8553/idsadmin** from the LAN because the admin interface is not exposed over reverse-proxy.

Step 2 Choose **Settings** > **IdS Trust**.

Step 3 Validate that the proxy cluster publisher node is listed on Download SP metadata page, and click **Next**.

Step 4 Validate that the IDP proxy is correctly displayed if configured on Upload IDP metadata page, and click **Next**.

Step 5 Initiate test SSO via all proxy cluster nodes from the Test SSO page and validate that all are successful. This requires client machine connectivity to reverse-proxy nodes.

Brute Force Attack Prevention Configuration

Finesse 12.6 ES02 and above authentication scripts actively prevent brute force attacks that can be used to guess the user password. The scripts do 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 number of failed requests, time interval, and blocking duration are configurable.

Attack Detection Parameters

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

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**. 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 a similar intrusion prevention system to add the blocked IP addresses to the IPtable or firewall rules.

Install and Configure Fail2ban

Fail2ban can be configured to monitor the `blocking.log` to identify the IP addresses that are blocked by OpenResty Nginx on detecting brute force attacks, and ban the IP addresses for a configurable duration. Do the following to install and configure Fail2ban on a CentOS reverse-proxy:

Procedure

Step 1 Install Fail2ban using yum

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

Step 2 Create a local jail

Jail configurations allow the administrator to configure various properties such as 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, and so on. Steps to add a custom configuration for banning IP addresses that are blocked from accessing the upstream servers are as follows:

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

```
cd /etc/fail2ban
```

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

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

c. Add the following 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
```

Step 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 follows:

a. Create filter.d/finesseban.conf

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

b. Add the following 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
```

Step 4 Start Fail2ban

Run the following 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`.

Troubleshoot

Troubleshoot SELinux

Procedure

Step 1 If OpenResty Nginx is not started by default or the Finesse Agent Desktop is not accessible, set SELinux to **permissive** mode with this command:

```
setenforce 0
```

Step 2 Try to restart OpenResty Nginx with the **`systemctl restart nginx`** command.

Step 3 All the violations will be available in **/var/log/messages** and **/var/log/audit/audit.log**.

Step 4 You are required to regenerate the **.te** file with allow rules for addressing those violations by executing any one of the following commands:

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

Step 5 Update the original **nginx.te** file present in the **selinux-nginx-rhel-master-modified/nginx** directory with the newly generated allow rules.

Step 6 Compile the **nginx.te** file by using the **`make`** command.

Step 7 The **nginx.pp** file is regenerated.

Step 8 Load the policy by using the **semodule** command.

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

Step 9 Change SELinux to **enforce** mode by using the **setenforce** command.

Step 10 Reboot the system.

Step 11 Repeat this procedure until all the violations are fixed.

Reverse-Proxy Configuration