Cisco Finesse 12.6 ES 02에 대한 VPN-Less 액세스를 위한 Nginx Reverse Proxy 구성

목차

소개

이 문서에서는 Cisco Finesse, Cisco CUIC(Unified Intelligence Center) 및 Cisco IdS(Identity Service)의 12.6 ES02 버전을 기반으로 VPN에 연결하지 않고 역방향 프록시를 사용하여 Cisco Finesse 데스크톱에 액세스하는 방법에 대해 설명합니다.

> **참고:** Nginx 설치 및 구성은 Cisco에서 지원하지 않습니다. 이 주제에 대한 질의는 Cisco 커뮤니티 포럼에서 다룰 수 있습니다.

> **참고:** VPN-Less의 ES02 구축의 경우 업그레이드 계획을 세우고 호환성 제한을 확인하려면 개별 구성 요소의 릴리스 정보를 참조하십시오.

사전 요구 사항

요구 사항

다음 주제에 대한 지식을 보유하고 있으면 유용합니다.

- Cisco UCCE(Unified Contact Center Enterprise) 릴리스
- Cisco Finesse
- Linux 관리
- 네트워크 관리 및 Linux 네트워크 관리

사용되는 구성 요소

이 문서의 정보는 다음 소프트웨어 및 하드웨어 버전을 기반으로 합니다.

- Finesse - 12.6 ES02
- CUIC - 12.6 ES02
- IdS - 12.6 ES02
- 컨택 센터(CC)용 UCCE/HCS(Hosted Collaboration Solution) - 11.6 이상
- PCCE(Packaged Contact Center Enterprise) - 12.0 이상

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 이해해야 합니다.

> **참고:** 이 문서에 제공된 컨피그레이션은 샘플 2000 사용자 UCCE 구축에 대해 CentOS 8.0에 구축된 Nginx reverse proxy(OpenResty)를 사용하여 구성, 강화 및 로드되었습니다. 이 문서에서는 성능 프로파일 참조 정보를 사용할 수 있습니다.

배경 정보

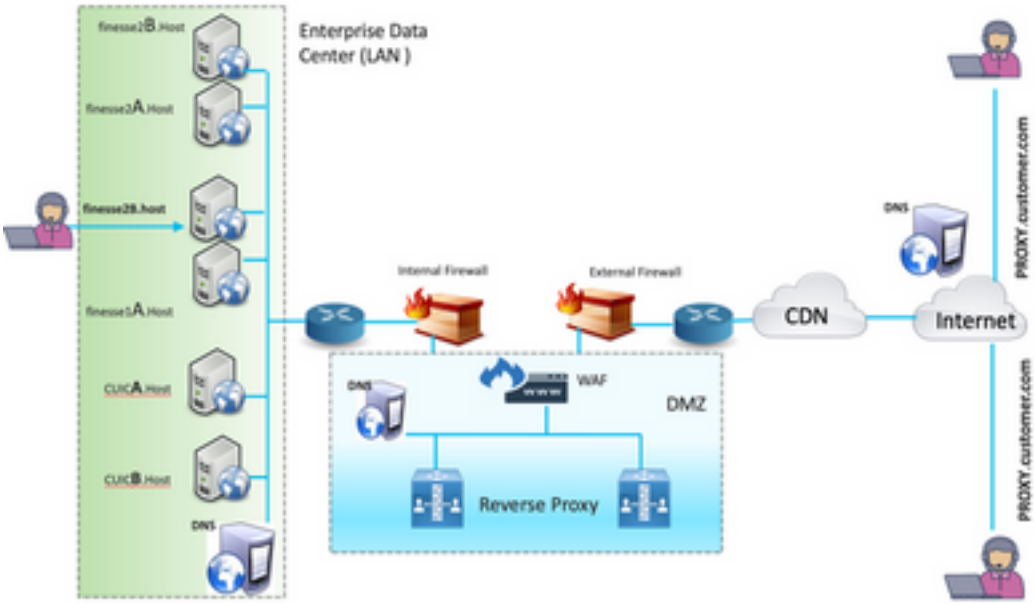
이 구축 모델은 UCCE 솔루션용 UCCE/PCCE 및 HCS에서 지원됩니다.

VPN에 연결하지 않고도 Cisco Finesse 데스크톱에 액세스할 수 있는 옵션으로 역방향 프록시를 구축할 수 있습니다(12.6 ES01에서 사용 가능). 이 기능은 상담원이 인터넷을 통해 어디서나 Finesse 데스크톱에 액세스할 수 있는 유연성을 제공합니다.

이 기능을 활성화하려면 리버스 프록시 쌍을 DMZ(Demilitarized Zone)에 구축해야 합니다.

리버스 프록시 구축에서는 미디어 액세스가 변경되지 않습니다. 미디어에 연결하기 위해 에이전트는 MRA(Mobile and Remote Access Solution)를 통한 Cisco Jabber를 사용하거나 PSTN(Public Switched Telephone Network) 또는 모바일 엔드포인트에서 UCCE의 Mobile Agent 기능을 사용할 수 있습니다. 이 다이어그램은 리버스 프록시 노드의 단일 HA(고가용성) 쌍을 통해 두 개의 Finesse 클러스터와 두 개의 CUIC 노드에 액세스할 때 네트워크 구축의 모양을 보여줍니다.

이 이미지에 표시된 대로 인터넷 상담원과 LAN에서 연결하는 상담원의 동시 액세스가 지원됩니다.



참고: 이 구축을 지원하려면 Nginx 대신 서드파티 프록시 선택 기준에 대한 기능 가이드를 참조하십시오.

- UCCE 12.6 기능 가이드 - VPN-Less 기능에 대한 기능 개요, 설계 및 **컨피그레이션 세부사항**을 제공합니다.
- UCCE 12.6 보안 가이드 - 리버스 프록시 구축을 위한 보안 컨피그레이션 지침을 제공합니다.

이 문서를 읽기 전에 기능 설명서 및 보안 설명서의 VPN-Less 섹션을 검토하는 것이 좋습니다.

ES02의 변경 사항

- 새로운 기능

이제 Finesse 수퍼바이저 기능이 리버스 프록시를 통해 지원됩니다.이제 CUIC RealTime 및 Historical 보고서는 프록시된 환경에서 Finesse 가젯을 통해 지원됩니다.

모든 요청/통신에 대한 인증 - Lua 지원 필요
모든 Finesse/CUIC/IM & Presence(IM&P) 요청은 데이터 센터에 들어가기 전에 프록시에서 인증됩니다.웹 소켓 및 라이브 데이터 소켓IO 연결도 제한되며 Finesse에 보안 요청을 성공적으로 수행한 클라이언트에서만 허용됩니다.무차별 대입 공격 감지 및 프록시에 대한 로깅. 이 기능은 Fail2Ban과 함께 사용하여 악성 IP 주소를 차단할 수 있습니다.

- 리버스 프록시 컨피그레이션을 위한 보안 향상 - Lua 지원 필요
  리버스 프록시와 업스트림 구성 요소(Finesse/IdS/CUIC/Livedata) 간의 TLS(Mutual Transport Layer Security) 인증Linux 설정.프록시 및 구성 요소 서버 요청에 대한 상호 SSL(Secure Sockets Layer) 신뢰 확인을 활성화합니다.

- DoS(Denial-of-Service)/DDoS(Distributed Denial-of-Service) 공격을 방지하기 위해 프록시 컨피그레이션에 대한 향상된 보안 - Lua 지원 필요
  시스템의 다양한 부분에 대한 향상된 Nginx 요청 속도 제한IpTables의 속도 제한.업스트림 구성 요소 서버를 요청하기 전에 정적 리소스 요청을 확인합니다.업스트림 구성 요소 서버에 도달하지 않는 더 가볍고 캐시할 수 있는 인증되지 않은 페이지입니다.

- 기타 기능 - Lua 지원 필요 자동 컨피그레이션을 지원하고 성능을 개선하기 위해 프록시에서 제공하는 CORS(Cross-Origin Resource Sharing) 응답을 자동으로 감지합니다.

ES01 기반 VPN-Less 컨피그레이션에 대한 업그레이드 참고 사항

- ES02 컨피그레이션에는 Lua를 지원하는 Nginx 설치가 필요합니다.
- 인증서 요구 사항 Nginx ES02 컨피그레이션이 업스트림 서버에 성공적으로 연결하기 전에 Cisco Finesse, CUIC 및 IdS에서 Nginx/OpenResty 호스트 인증서를 Tomcat 트러스트 저장소에 추가하고 다시 시작해야 합니다.ES02 기반 구성을 사용하려면 Nginx 서버에서 Cisco Finesse, CUIC 및 IdS 업스트림 서버 인증서를 구성해야 합니다.

  참고: ES02 Nginx 컨피그레이션을 설치하기 전에 기존 ES01 기반 Nginx 컨피그레이션을 제거하는 것이 좋습니다.

  참고: ES02 컨피그레이션 스크립트에는 Cisco Finesse, CUIC 및 IdS에 해당하는 ES02 COP를 설치해야 합니다.

인증

Finesse 12.6 ES02는 프록시에서 인증을 도입합니다. SSO(Single Sign On) 및 비 SSO 구축에서는 인증이 지원됩니다.

인증은 프록시에서 수락한 모든 요청 및 프로토콜에 대해 시행되며, 이 프로토콜은 업스트림 구성 요소 서버로 전달되기 전에 적용됩니다. 여기서 구성 요소 서버에서 로컬로 시행하는 인증도 수행됩니다. 모든 인증에서는 일반적인 Finesse 로그인 자격 증명을 사용하여 요청을 인증합니다.

인증 및 사후 연결을 위해 XMPP(Extensible Messaging and Presence Protocol)와 같은 애플리케이

션 프로토콜을 사용하는 웹 소켓과 같은 영구 연결은 소켓 연결을 설정하기 전에 애플리케이션 인증이 성공한 IP 주소를 검증하여 프록시에서 인증됩니다.

비 SSO 인증

비SSO 인증에는 추가 컨피그레이션이 필요하지 않으며 필요한 스크립트 교체가 완료되면 즉시 Nginx 컨피그레이션 스크립트를 사용할 수 있습니다. 인증은 Finesse에 로그인하는 데 사용되는 사용자 이름과 비밀번호를 사용합니다. 모든 엔드포인트에 대한 액세스는 Finesse 인증 서비스로 검증됩니다.

유효한 사용자 목록은 프록시에서 로컬로 캐시됩니다(15분마다 캐시를 업데이트). 이 목록은 요청에서 사용자를 확인하는 데 사용됩니다. 사용자 자격 증명이 구성된 Finesse URI에 요청을 전달하여 유효성을 검사하고 있으며, 그런 다음 자격 증명 해시가 로컬에 캐시되어(15분 캐시) 새 요청을 로컬로 인증합니다. 사용자 이름 또는 비밀번호가 변경되면 15분 후에만 적용됩니다.

SSO 인증

SSO 인증에서는 관리자가 구성 파일 내의 Nginx 서버에서 IdS 토큰 암호화 키를 구성해야 합니다. IdS 토큰 암호화 키는 **show ids secret** CLI 명령. SSO 인증이 작동하기 전에 관리자가 스크립트에서 수행해야 하는 #Must-change 교체 중 하나로 키를 구성해야 합니다.

프록시 확인이 IdS에 대해 작동하려면 수행할 IdS SAML 컨피그레이션에 대한 SSO 사용 설명서를 참조하십시오.

SSO 인증이 구성되면 유효한 토큰 쌍을 사용하여 시스템의 엔드포인트에 액세스할 수 있습니다. 프록시 컨피그레이션은 IdS에 대한 토큰 검색 요청을 가로채거나 유효한 토큰을 해독한 다음 추가 검증을 위해 이를 로컬로 캐싱하여 자격 증명을 검증합니다.

웹 소켓 연결에 대한 인증

사용자 지정 헤더는 브라우저의 기본 websocket 구현에서 지원되지 않으므로 표준 권한 부여 헤더로 웹 소켓 연결을 인증할 수 없습니다. 페이로드에 포함된 인증 정보가 웹 소켓 연결 설정을 방지하지 않는 애플리케이션 수준 인증 프로토콜입니다. 따라서 악의적인 엔터티는 시스템을 압도하기 위해 수많은 연결을 생성하여 DOS 또는 DDOS 공격을 렌더링할 수 있습니다.

이러한 가능성을 완화하기 위해 제공된 nginx 역방향 프록시 컨피그레이션에서는 websocket 연결을 설정하기 전에 인증된 REST 요청을 성공적으로 수행한 IP 주소로부터 웹 소켓 연결을 수락할 수 있도록 특정 검사를 수행합니다. 즉, REST 요청이 발행되기 전에 웹 소켓 연결을 생성하려고 시도하는 클라이언트는 권한 부여 실패 오류를 가져오고 지원되는 사용 시나리오가 아닙니다.

무작위 대입 공격 방지

Finesse 12.6 ES02 인증 스크립트는 사용자 비밀번호를 추측하는 데 사용할 수 있는 무작위 대입 공격을 능동적으로 방지합니다. 이렇게 하려면 짧은 시간에 몇 번의 실패 시도 후에 서비스에 액세스하는 데 사용되는 IP 주소를 차단합니다. 이러한 요청은 **418 클라이언트 오류**로 거부됩니다. 차단된 IP 주소의 세부 정보는 **<nginx-install-directory>/logs/blocking.log** 및 **<nginx-install-directory>/logs/error.log** 파일**에서 액세스할** 수 있습니다**.**

실패한 요청 수, 시간 간격 및 차단 기간을 구성할 수 있습니다. 구성은 **<nginx-install-directory>/conf/conf.d/maps.conf** 파일**에** 있습니다.

```
## 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;
}
```

로깅

```
grep -r "IP is already blocked." error.log
========================================

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"

tail -f blocking.log
====================

2021/10/29 17:30:59 [error] 939738#939738: *1857 [lua] content_by_lua(rest_cache:189 2:
[10.70.235.30] will be blocked for [ 30 minutes ] for exceeding retry limit., client:
10.70.235.30, server: saproxy.cisco.com, request: "GET /finesse/api/SystemInfo HTTP/1.1", host:
"saproxy.cisco.com:8445"
```

고객은 Fail2Ban 또는 이와 유사한 방식으로 통합하여 IPtable/방화벽 규칙에 금지를 추가하는 것이 좋습니다.

정적 리소스 URL 확인

인증되지 않은 방식으로 액세스할 수 있는 모든 유효한 엔드포인트는 ES02 스크립트에서 추적됩니다.

이러한 인증되지 않은 경로에 대한 요청은 업스트림 서버로 요청을 보내지 않고 잘못된 URI가 요청되면 적극적으로 거부됩니다.

CORS 헤더 캐시

첫 번째 옵션 요청이 성공하면 응답 헤더 **access-control-allow-headers, access-control-allow-origin**, **access-control-allow-methods, access-control-expose-headers** 및 **access-control-allow-credentials**가 프록시에 5분 동안 캐시됩니다. 이러한 헤더는 각 업스트림 서버에 대해 캐시됩니다.

구성

이 문서에서는 Nginx를 Finesse VPN-Less 액세스를 활성화하는 데 사용할 리버스 프록시로 구성하는 방법에 대해 설명합니다. 제공된 지침을 확인하는 데 사용되는 UCCE 솔루션 구성 요소, 프록시 및 OS 버전이 제공됩니다. 관련 지침은 선택한 OS/프록시에 맞게 조정되어야 합니다.

- 사용된 Nginx 버전 - OpenResty 1.19.9.1
- 구성에 사용되는 OS - CentOS 8.0

참고: 설명된 Nginx 컨피그레이션은 Finesse Release 12.6(1)ES2 소프트웨어 다운로드 페이지에서 다운로드할 수 있습니다.

DMZ에서 OpenResty를 역방향 프록시로 설치

이 섹션에서는 OpenRestore 기반 프록시 설치 단계를 자세히 설명합니다. 리버스 프록시는 일반적으로 앞서 언급한 구축 다이어그램에 표시된 대로 DMZ(network demilitarized zone)에서 전용 디바이스로 구성됩니다.

1. 필요한 하드웨어 사양을 사용하여 **원하는 OS**를 설치합니다. 커널과 IPv4 매개 변수 조정은 선택한 OS에 따라 다를 수 있으며, 선택한 OS 버전이 다른 경우 이러한 측면을 다시 확인하는 것이 좋습니다.
2. 2개의 네트워크 인터페이스를 구성합니다. 인터넷 클라이언트에서 공용 액세스를 위해서는 인터페이스 하나가 필요하고 내부 네트워크의 서버와 통신하려면 다른 인터페이스가 필요합니다.
3. OpenResty를 설치합니다.

Nginx 1.19+를 기반으로 하고 Lua를 지원하는 경우 이 용도로 Nginx의 모든 종류를 사용할 수 있습니다.

- Nginx 플러스
- Nginx 오픈 소스(Nginx 오픈 소스를 OpenResty 기반 Lua 모듈과 함께 컴파일해야 사용할 수 있음)
- 오픈 복원
- GetPageSpeed Extras

참고: 제공된 컨피그레이션은 OpenResty 1.19에서 테스트되었으며 간단한 업데이트만 있는 다른 배포에서 작동할 것으로 예상됩니다(있는 경우).

OpenRestore 설치

1. OpenResty를 설치합니다. OpenRestore Linux 패키지를 참조하십시오. OpenRestore 설치의 일부로 Nginx가 이 위치에 설치되고 ***~/.bashrc*** 파일에 추가하여 ***PATH*** 변수에 OpenRestore 경로*를* 추가합니다.
   ```
   export PATH=/usr/local/openresty/bin:$PATH
   ```
2. Nginx 시작/중지 Nginx를 시작하려면 **openresty**.Nginx를 중지하려면 **openresty -s stop**.

Nginx 구성

이 컨피그레이션은 OpenResty 기반 Nginx 설치에 대해 설명합니다. OpenResty의 기본 디렉토리는 다음과 같습니다.

- <nginx 설치 디렉토리> = /usr/local/openresty/nginx

- <OpenResty-install-directory> = /usr/local/openresty

1. Nginx에 대한 리버스 프록시 컨피그레이션이 포함된 Finesse Release 12.6(1)ES2 소프트웨어 다운로드 페이지(12.6-ES02-reverse-proxy-config.zip)에서 파일을 다운로드하고 추출합니다.
2. 추출된 역방향 프록시 구성 디렉토리에서 nginx.conf, nginx/conf.d/및 nginx/html/을 복사하여 **<nginx-install-directory>/conf, <nginx-install-directory>/conf/conf.d/** 및 **<nginx-install-directory>/html/**로 복사합니다.
3. **<nginx-install-directory>** 내부의 추출된 역방향 프록시 컨피그레이션 디렉토리에서 nginx**/lua** 디렉터리를 **복사합니다.**
4. lualib의 내용**을 <Openresty-install-directory>/lualib/resty에 복사합니다**.
5. nginx/logtate**/saproxy** 파일을 **<nginx-install-directory>/logtate/**폴더에 복사하여 nginx 로그 순환**을 구성합니다**. Nginx 기본값을 사용하지 않는 경우 올바른 로그 디렉토리를 가리키도록 파일 내용을 수정합니다.
6. Nginx는 권한이 없는 전용 서비스 계정으로 실행되어야 합니다. 이 계정은 잠겨 있고 잘못된 셸(또는 선택한 OS에 적용 가능)이 있어야 합니다.
7. **html** 및 **conf.d라는** 추출된 폴더 아래의 파일에서 "**반드시** 변경해야 **하는**" 문자열을 찾아 표시된 값을 적절한 항목으로 교체합니다.
8. 모든 필수 교체 작업이 완료되었는지 확인합니다. 이 작업은 컨피그레이션 파일의 **Must-change** 코멘트와 함께 설명합니다.
9. CUIC 및 Finesse에 대해 구성된 캐시 디렉토리가 이러한 임시 디렉토리와 함께 **<nginx-install-directory>/cache**에 작성되었는지 확인합니다. <nginx 설치 디렉토리>/캐시/client_temp<nginx 설치 디렉토리>/캐시/proxy_temp

   참고: 제공된 컨피그레이션은 샘플 2000 구축을 위한 것이며 대규모 구축을 위해 적절하게 확장되어야 합니다.

Nginx 캐시 구성

기본적으로 프록시 캐시 경로는 파일 시스템에 저장됩니다. 여기에 표시된 것처럼 tmpfs에서 캐시 위치를 생성하여 메모리 내 드라이브로 변경하는 것이 좋습니다.

1. /home 아래에 다른 프록시 캐시 경로에 대한 디렉토리를 만듭니다. 예를 들어, 기본 Finesse에 대해 이러한 디렉토리를 만들어야 합니다. 보조 Finesse 및 CUIC 서버에서도 동일한 단계를 수행해야 합니다.

```
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
```
참고: 구성에 추가된 새 Finesse 클러스터 각각에 대해 클라이언트 및 proxy_temp 캐시를 1GB씩 늘립니다.

2. 명령을 사용하여 새 마운트 지점 마운트 **mount -av**.
3. 파일 시스템이 새 마운트 지점을 마운트했는지 확인 **df -h** 명령을 실행합니다.
4. Finesse 및 CUIC 캐시 컨피그레이션 파일에서 proxy_cache_path 위치를 변경합니다. 예를 들어 Finesse 1의 경로를 변경하려면 **<nginx-install-directory>conf/conf.d/finesse/cache로 이동하여** 기존 캐시 위치**/usr/local/openresty/nginx/cache/finesse25/**를 새로 생성된 파일 시스템 위치**/home/primaryFinesse로 변경합니다.**
```
##Must-change /usr/local/openresty/nginx/cache/
location would change depending on folder extraction ##
Nginx config file to cache the desktop/shindig and notification service related static
files.
proxy_cache_path /home/primaryFinesse/desktop levels=1:2 use_temp_path=on
keys_zone=desktop_cache_primary: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_primary: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_primary: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:10m max_size=1500m inactive=40m
use_temp_path=off;
```
5. Finesse 보조 및 CUIC 서버에 대해 동일한 단계를 수행합니다.

참고: 모든 이전 단계에서 생성된 모든 MPFS 드라이브 크기의 합계가 구축의 최종 메모리 크기 조정에 추가되어야 합니다. 이러한 드라이브는 애플리케이션에 디스크처럼 보이도록 구성된 메모리 블록으로 메모리 공간을 최대한 많이 소비하기 때문입니다.

SSL 인증서 구성

자체 서명 인증서 사용 - 테스트 배포

자체 서명 인증서는 리버스 프록시를 프로덕션 환경으로 롤아웃할 준비가 될 때까지 사용해야 합니다. 프로덕션 구축에서는 CA(Certificate Authority) 서명 인증서만 사용합니다.

1. SSL 폴더 콘텐츠에 대한 Nginx 인증서를 생성합니다. 인증서를 생성하기 전에 /usr/local/openresty/nginx 아래에 **ssl**이라는 폴더**를** 만들어야 합니다. 두 개의 호스트 이름(동일한 프록시 서버)을 사용하여 Finesse node1 및 Finesse node2에 액세스하므로 이러한 명령의 도움말을 사용하여 두 개의 인증서(**<reverseproxy_primary_fqdn>**용 및 <reverseproxy**_secondary_fqdn>**에 대한 인증서**를** 생성해야 합니다. **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** (으)로 호스트 이름 전달: <reverseproxy_primary_fqdn>**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** (호스트 이름 전달:<reverseproxy_secondary_fqdn>)인증서 경로가 **etc/nginx/ssl/nginx.crt** 및 **/usr/local/openresty/nginx/ssl/nginxnode2.crt**인지 확인합니다. Finesse Nginx 구성 파일에 이미 구성되어 있습니다.
2. 개인 키 **400(r—)**의 권한을 **변경합니다**.

3. Nginx 서버가 수신 대기하도록 구성된 포트에 대응하도록 방화벽에서 통신을 활성화하려면 역방향 프록시에 방화벽/iptables를 구성합니다.
4. 역방향 프록시 서버의 **/etc/hosts** 항목 아래에 Finesse, IdS 및 CUIC의 IP 주소 및 호스트 이름을 추가합니다.
5. Nginx 호스트를 리버스 프록시로 구성하려면 구성 요소 서버에서 수행할 구성에 대한 솔루션 기능 가이드를 참조하십시오.

참고: 제공된 컨피그레이션은 샘플 2000 구축을 위한 것이며 대규모 구축을 위해 적절하게 확장되어야 합니다.

CA 서명 인증서 사용 - 프로덕션 구축

CA 서명 인증서는 다음 단계를 통해 역방향 프록시에 설치할 수 있습니다.

1. CSR(Certificate Signing Request)을 생성합니다. CSR 및 개인 키를 생성하려면 **openssl req -new -newkey rsa:4096 -keyout nginx.key -out nginx.csr** 프록시에 로그인한 후 프롬프트에 따라 세부사항을 입력합니다. 그러면 강도 4096비트의 CSR(예제의 nginx.csr) 및 RSA 개인 키(예제의 nginx.key)가 생성됩니다.예를 들면 다음과 같습니다.

```
[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.comEmail 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
```

   PEM 암호 구문을 기록해 둡니다. 구축 중에 개인 키를 해독하는 데 사용됩니다.
2. CA에서 서명된 인증서를 가져옵니다. CSR을 인증 기관에 보내고 서명된 인증서를 가져옵니다.
3. 인증서 및 키를 구축합니다. 에서 첫 번째 단계의 일부로 생성된 키를 해독합니다. **openssl rsa -in nginx.key -out nginx_decrypted.key** 명령을 실행합니다. CA 서명 인증서 및 암호 해독된 키를 역방향 프록시 시스템의 폴더(예에서 /usr/local/openresty/nginx/ssl)에 배치합니다. Nginx 컨피그레이션의 인증서와 관련된 SSL 컨피그레이션을 업데이트/추가합니다.

```
server {
    server_name <proxy-name>;
    listen 8445 ssl reuseport http2;
```

```
        listen [::]:8445 ssl reuseport http2;
        .....
        ssl_certificate /usr/local/openresty/nginx/ssl/ca_signed_cert.crt;
        ssl_certificate_key /usr/local/openresty/nginx/ssl/nginx_decrypted.key;
        .....
    }
```

4. 인증서에 대한 권한을 구성합니다. 입력 **chmod 400 /usr/local/openresty/nginx/ssl/ca_signed_cert.crt** 및 **chmod 400 /usr/local/openresty/nginx/ssl/nginx_decrypted.key**- 인증서가 읽기 전용 권한을 가지며 소유자로 제한됩니다.
5. Nginx를 다시 로드합니다.

사용자 지정 Diffie-Hellman 매개 변수 사용

다음 명령을 사용하여 사용자 지정 Diffie-Hellman 매개변수를 생성합니다.

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

새 매개변수를 사용하도록 서버 컨피그레이션을 변경합니다.

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

OCSP 스테이플링이 활성화되었는지 확인 - 인증서 해지 확인

> **참고:** 이를 활성화하려면 서버가 CA 서명 인증서를 사용해야 하며, 서버는 인증서를 서명한 CA에 액세스할 수 있어야 합니다.

구성에 추가:

```
server {
    .....
    ssl_stapling on;
    ssl_stapling_verify on;
    .....
}
```

Nginx 구성

보안을 적용하고 성능을 제공하기 위해 기본 Nginx 구성 파일 (**/usr/local/openresty/nginx/conf/nginx.conf**)을 수정해야 합니다. 이 콘텐츠는 Nginx 설치에서 만든 기본 구성 파일을 수정하는 데 사용해야 합니다.

```
# 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. 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;
```

역방향 프록시 포트 구성

기본적으로 Nginx 컨피그레이션은 포트 8445에서 Finesse 요청을 수신합니다. 한 번에 Finesse 요청을 지원하기 위해 리버스 프록시에서 포트 하나만 활성화할 수 있습니다(예: 8445). 포트 443을 지원해야 하는 경우 443에서 수신을 활성화하고 8445에서 수신을 비활성화하려면 **<nginx-install-directory>conf/conf.d/finesse.conf** 파일을 편집합니다.

역방향 프록시와 업스트림 구성 요소 간의 상호 TLS 인증 구성

ES02에서는 기본적으로 모든 업스트림 구성 요소가 allowed-hosts CLI utils **system reverse-proxy allowed-hosts add <proxy-host>** 명령의 일부로 추가된 모든 호스트**를 검증하도록** 구성됩니다. 따라서 역방향 프록시 호스트와 업스트림 구성 요소(Finesse/IdS/CUIC/Livedata) 간에 성공적으로 통신하려면 역방향 프록시 인증서를 모든 업스트림 구성 요소의 tomcat 트러스트 저장소에 업로드해야 합니다. 인증서가 업로드되면 노드를 다시 시작합니다.

역방향 프록시에 의한 업스트림 서버 인증서 검증은 선택 사항이며 기본적으로 비활성화되어 있습니다. 역방향 프록시와 업스트림 호스트 간에 전체 TLS 상호 인증을 수행하려면 이 컨피그레이션을 **ssl.conf** 및 **ssl2.conf** 파일에서 주석**을** 해제해야 합니다.

```
#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** 파일은 연결된 모든 업스트림 인증서 엔트리를 포함해야 합니다.

캐시 지우기

리버스 프록시 캐시는 명령을 실행합니다.

표준 지침

이 섹션에서는 Nginx를 프록시 서버로 설정할 때 따라야 할 표준 지침에 대해 간략하게 설명합니다.

이러한 지침은 Center for Internet Security에서 파생됩니다. 각 지침에 대한 자세한 내용은 같은 항목을 참조하십시오.

1. 항상 최신 OpenResty 및 OpenSSL 버전을 사용하는 것이 좋습니다.
2. 별도의 디스크 마운트에 Nginx를 설치하는 것이 좋습니다.
3. Nginx 프로세스 ID는 루트 사용자(또는 선택한 OS에 적용 가능)가 소유해야 하며 권한 **644(rw—)** 또는 엄격한 권한이 있어야 합니다.
4. Nginx는 알 수 없는 호스트에 대한 요청을 차단해야 합니다. 각 서버 블록에 명시적으로 정의된 server_name 지시어가 포함되어 있는지 확인합니다. 확인하려면 nginx.conf 및 **nginx/conf.d** 디렉토리에서 모든 서버 블록을 검색하고 모든 서버 블록에 server_name이 포함되어 있는지 확인합니다.
5. Nginx는 인증된 포트에서만 수신해야 합니다. nginx.**conf** 및 **nginx/conf.d** 디렉토리에서 모든 서버 블록을 검색하고 지시어 수신 대기 를 확인하여 인증된 포트만 수신 대기 상태인지 확인합니다.
6. Cisco Finesse는 HTTP를 지원하지 않으므로 프록시 서버 HTTP 포트도 차단하는 것이 좋습니다.
7. Nginx SSL 프로토콜은 TLS 1.2여야 합니다. 레거시 SSL 프로토콜에 대한 지원을 제거해야 합니다. 또한 취약한 SSL 암호를 비활성화해야 합니다.
8. Nginx 오류 및 액세스 로그를 원격 syslog 서버로 전송하는 것이 좋습니다.
9. 웹 애플리케이션 방화벽으로 작동하는 **mod_security** 모듈을 설치하는 것이 좋습니다. 자세한 내용은 ModSecurity 설명서를 참조하십시오. Nginx 부하가 **mod_security** 모듈 내에서 확인되지 않았습니다.

매핑 파일 구성

Finesse 데스크톱의 역방향 프록시 구축에는 외부에 표시되는 호스트 이름/포트 조합 목록과 Finesse, IdS 및 CUIC 서버에서 사용하는 실제 서버 이름 및 포트에 대한 매핑을 구성하기 위한 매핑 파일이 필요합니다. 내부 서버에 구성된 이 매핑 파일은 인터넷을 통해 연결된 클라이언트가 인터넷에서 사용되는 필수 호스트 및 포트로 리디렉션되도록 하는 핵심 컨피그레이션입니다.

구성 요소 서버에서 액세스할 수 있는 웹 서버에 매핑 파일을 구축해야 하며 구축이 작동하려면 해당 URI를 구성해야 합니다. 네트워크 내에서 사용 가능한 전용 웹 서버를 사용하여 매핑 파일을 구성하는 것이 좋습니다. 이러한 서버를 사용할 수 없는 경우 역방향 프록시를 대신 사용할 수 있습니다. 그러면 네트워크 내에서 프록시에 액세스할 수 있어야 하며 DMZ에 무단 액세스를 수행할 수 있는 외부 클라이언트에 정보를 노출할 위험이 있습니다. 다음 섹션에서는 이 작업을 수행하는 방법을 자세히 설명합니다.

모든 구성 요소 서버에서 매핑 파일 URI를 구성하는 정확한 단계 및 매핑 파일 데이터를 만드는 방법에 대한 자세한 내용은 기능 가이드를 참조하십시오.

역방향 프록시를 매핑 파일 서버로 사용

이러한 단계는 역방향 프록시가 프록시 매핑 파일 호스트로도 사용되는 경우에만 필요합니다.

1. Finesse/CUIC 및 IdS 호스트에서 사용하는 도메인 컨트롤러에서 리버스 프록시 호스트 이름을 구성하여 해당 IP 주소를 확인할 수 있습니다.
2. cmplatform의 tomcat-trust 아래에 있는 두 노드에서 생성된 Nginx 서명 인증서를 업로드하고 서버를 다시 시작합니다.
3. <NGINX_HOME>/html/proxymap.txt에서 Must-change 값을 업데이트합니다.
4. 를 사용하여 Nginx 구성 다시 로드 **nginx -s reload** 명령을 실행합니다.
5. 를 사용하여 다른 네트워크 호스트에서 구성 파일에 액세스할 수 있는지 확인합니다. **curl** 명령을 실행합니다.

CentOS 8 커널 강화

선택한 운영 체제가 CentOS 8인 경우, 프록시 호스팅을 위해 전용 서버를 사용하는 설치에 대해 이러한 sysctl 구성을 사용하여 커널 강화/조정을 수행하는 것이 좋습니다.

```
## 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
```

권장 사항을 변경한 후 재부팅하는 것이 좋습니다.

IPtables 강화

IPtables는 시스템 관리자가 Linux 커널 방화벽에서 제공하는 IPv4 및 IPv6 테이블, 체인 및 규칙을 구성할 수 있는 애플리케이션입니다.

이러한 IPtables 규칙은 Linux 커널 방화벽에서 액세스를 제한하여 무작위 대입 공격으로부터 프록시 애플리케이션을 보호하도록 구성됩니다.

컨피그레이션의 코멘트는 규칙을 사용하여 어떤 서비스가 속도 제한을 받고 있는지 나타냅니다.

> **참고:** 관리자가 다른 포트를 사용하거나 동일한 포트를 사용하여 여러 서버에 대한 액세스를 확장할 경우 이러한 번호에 따라 적절한 크기 조정을 수행해야 합니다.

```
## 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-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
```

이러한 규칙은 **/etc/sysconfig/iptables**를 수동으로 편집하거나 **iptables.conf**와 같은 파일에 컨피그레이션을 저장하고 cat iptables.conf >>/etc/sysconfig/iptables를 실행하여 규칙을 적용할 수 있습니다.

규칙을 적용한 후 IPtables 서비스를 다시 시작해야 합니다. 입력 **systemctl restart iptables** IPtables 서

비스를 재시작합니다.

클라이언트 연결 제한

이전 IPtables 컨피그레이션 외에도 프록시를 사용하는 클라이언트의 주소 범위를 알고 있는 설치는 이 지식을 사용하여 프록시 액세스 규칙을 보호하는 것이 좋습니다. 이는 온라인 보안과 관련하여 좀 더 느슨한 규칙을 가지고 있는 국가의 IP 주소 범위에 자주 생성되는 악성 네트워크의 봇넷으로부터 프록시를 보호하는 데 있어 큰 비용을 지불할 수 있습니다. 따라서 액세스 패턴이 확실하다면 IP 주소 범위를 국가/주 또는 ISP 기반 IP 범위로 제한하는 것이 좋습니다.

클라이언트 연결 차단

또한 IP 주소 또는 IP 주소 범위에서 공격을 식별하여 특정 주소 범위를 차단하는 방법을 아는 것이 유용합니다. 이러한 경우 해당 IP 주소의 요청을 iptable 규칙으로 차단할 수 **있습니다**.

고유 IP 주소 차단

여러 고유 IP 주소를 차단하려면 각 IP 주소의 IPTables 컨피그레이션 파일에 라인을 추가합니다.

예를 들어 주소 **192.0.2.3** 및 **192.0.2.4**를 **차단하려면 다음을** 입력합니다.

```
iptables -A INPUT -s 192.0.2.3 -j DROP iptables -A INPUT -s 192.0.2.4 - j DROP.
```

IP 주소 범위 차단

한 범위의 여러 IP 주소를 차단하고 IP 범위의 IPTables 컨피그레이션 파일에 한 줄을 추가합니다.

예를 들어 192.0.2.3에서 192.0.2.35으로 주소를 차단하려면 다음을 입력합니다.

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

서브넷의 모든 IP 주소 차단

IP 주소 범위에 대해 클래스 없는 도메인 간 라우팅 표기법을 사용하여 IPTables 컨피그레이션 파일에 단일 라인을 추가하여 전체 서브넷의 모든 IP 주소를 차단합니다. 예를 들어 모든 클래스 **C 주소**를 차단하려면 다음을 입력합니다.

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

SELinux

SELinux는 Linux OS에 향상된 기능으로 통합된 플랫폼 보안 프레임워크입니다. OpenResty를 실행하는 SELinux 정책을 설치하고 추가하는 절차는 다음에 제공됩니다.

1. 프로세스를 **openresty -s stop** 명령을 실행합니다.
2. 를 사용하여 nginx 서버를 구성하고 시작/중지 **systemctl** 명령을 사용하여 부팅 중에 OpenRestore 프로세스가 자동으로 시작됩니다. 이러한 명령을 루트 사용자로 입력합니다. /usr/lib**/systemd/system으로 이동합니다**.openresty**.service**라는 파일**을 엽니다**.PIDFile 위치에 따라 파일의 내용**을** 업데이트합니다.
   ```
   [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
```

루트 사용자로 **sudo systemctl enable openresty.**를 사용하여 OpenRestore 서비스 시작/중지 **systemctl start openresty / systemctl stop openresty** 명령을 실행하여 프로세스가 루트 사용자로 시작/중지되었는지 확인합니다.

1. **Selinux 설치** 기본적으로 일부 SELinux 패키지만 CentOs에 설치됩니다.SELinux 정책**을** 생성하려면 policyCoreutils-devel 패키지 및 해당 종속성을 설치해야 합니다.policycorutils-devel을 설치하려면 이 명령**을 입력합니다.**

```
yum install policycoreutils-devel
```

패키지를 설치한 후 **sepolicy** 명령이 작동합니다.

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

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

SELinux Policy Inspection Tool
```

2. **SElinux 사용자로 새 Linux 사용자 및 맵 생성**입력 **semanage login -l** Linux 사용자와 SELinux 사용자 간의 매핑을 보려면

```
[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       *
```

루트로 SELinux **user_u** 사용자에 매핑된 새 Linux 사용자(**nginx** 사용자)를 생성합니다.

```
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.
```

nginxuser와 **user_u** 간의 매핑을 **보려면** root로 다음 명령을 입력합니다.

```
[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       *
```

SELinux **__default__**기본적으로 SELinux **unemoted_u** 사용자에 매핑됩니다. **user_u**를 기본적으로 다음 명령으로 제한해야 합니다.

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

명령이 제대로 작동하는지 확인하려면 **semanage login -l**. 다음과 같은 출력이 생성됩니다

.

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

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

nginx.conf를 수정하고 nginxuser에 대한 소유권 변경을 수행합니다.입력 **chown -R nginxuser:nginxuser** * <Openresty-**install-directory>** 디렉토리**에** 있습니다.nginx**.conf** 파일을 수정하여 작업자 프로세스를 실행하기 위한 사용자로 nginxuser를 포함합니다.

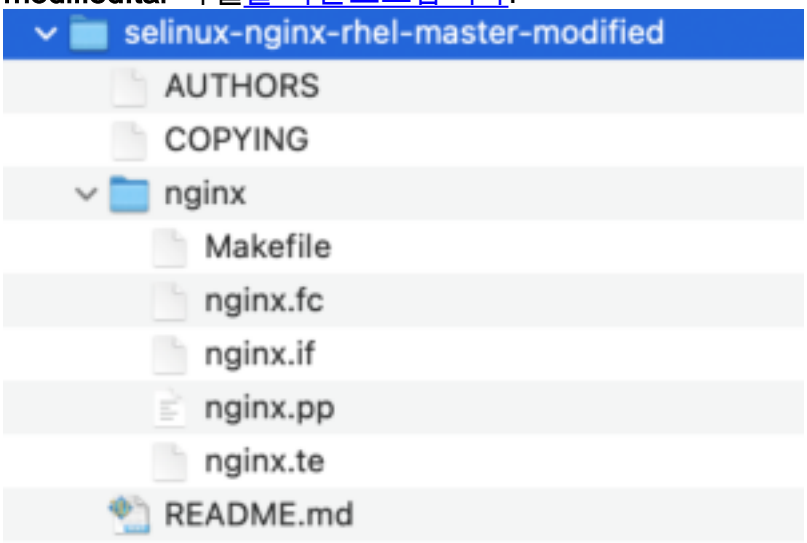
```
........

user nginxuser nginxuser;

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

Nginx에 대한 SELinux 정책 쓰기

1. Nginx에 대한 새 기본 사용자 지정 정책을 생성하는 대신 **sepolicy generate --init /usr/bin/nginx** 기존 정책으로 시작하는 것이 좋습니다.
2. 제공된 URL에서 다운로드한 nginx**.fc** 파일(File Contexts 파일) 및 **nginx.te**(Type Enforcement 파일) 파일이 역방향 프록시 사용에 맞게 수정되었습니다.
3. 이 수정된 버전은 특정 활용 사례에 대해 수정되었으므로 참조로 사용할 수 있습니다.
4. Finesse 12.6 ES02 파일 소프트웨어 다운로드 페이지에서 **selinux-nginx-rhel-master-modified.tar** 파일을 다운로드합니다.



5. .tar 파일을 추출하고 **nginx** 디렉토리**로** 이동합니다.
6. **.fc** 파일을 열고 nginx 설치 프로그램, 캐시 및 pid 파일의 필수 파일 경로를 확인합니다.
7. 컨피그레이션을 **make** 명령을 실행합니다.
8. nginx**.pp** 파일이 생성됩니다.
9. 정책을 **semodule** 명령을 실행합니다.
   ```
   semodule -i nginx.pp
   ```
10. /root로 **이동하여 touch /.autorelabel**이라는 빈 파일을 **만듭니다**.
11. 시스템을 재부팅합니다.
12. 정책이 성공적으로 로드되었는지 확인하려면 이 명령을 입력합니다.
    ```
    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. Nginx는 위반 없이 실행해야 합니다. (위반은 /var/**log/messages** 및 **/var/log/audit/audit.log**에서 사용할 수 있습니다.)
14. Nginx 상태를 확인하려면 이 명령을 입력합니다.

```
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. 이제 Finesse 에이전트/수퍼바이저 데스크톱에 액세스할 수 있어야 합니다.

다음을 확인합니다.

이 섹션을 사용하여 컨피그레이션이 제대로 작동하는지 확인합니다.

Finesse

1. https://<reverseproxy:port>/finesse/api/SystemInfo를 요청합니다. 연결할 수 있는지 확인합니다.
2. <primaryNode> 및 **<secondaryNode>**의 **<host>** 값은 유효한 역방향 프록시 호스트 이름입니다. Finesse 호스트 이름이 아니어야 합니다.

CUIC 및 라이브 데이터

1. Finesse 호스트 이름이 역방향 프록시 호스트 이름 대신 응답에 표시되는 경우 Finesse 12.6 UCCE 기능 설명서의 "VPN-Less Access to Finesse Desktop"의 "네트워크 변환 데이터 채우기" 섹션에 설명된 대로 프록시 매핑 컨피그레이션 및 허용된 호스트가 Finesse 서버에 올바르게 추가되었는지 확인합니다.
2. LiveData 가젯이 Finesse Desktop에서 제대로 로드되면 CUIC 및 LiveData 프록시 컨피그레이션이 적합합니다.
3. CUIC 및 LiveData 컨피그레이션의 유효성을 검사하려면 DMZ에서 이러한 URL에 대한 HTTP 요청을 만들고 해당 URL에 연결할 수 있는지 확인합니다.
   https://<reverseproxy:cuic_port>/cuic/rest/정보
   https://<reverseproxy:ldweb_port>/livedata/securityhttps://<reverseproxy:ldsocketio_port>/보

안

ID

IdS 컨피그레이션을 검증하려면 다음 단계를 수행합니다.

1. 관리자 인터페이스가 역방향 프록시를 통해 노출되지 않으므로 LAN에서 **https://<ids_LAN_host:ids_port>:8553/idadmin**의 IdSAdmin 인터페이스에 로그인합니다.
2. Settings(**설정) > IdS Trust(IDs 신뢰)를 선택합니다**.
3. 프록시 클러스터 게시자 노드가 Download SP metadata(SP 메타데이터 다운로드) 페이지에 나열되어 있는지 확인하고 Next(**다음**)를 클릭합니다.
4. Upload IDP metadata(IDP 메타데이터 업로드) 페이지에서 구성된 경우 IDP 프록시가 올바르게 표시되는지 확인하고 **Next(다음)**를 클릭합니다.
5. Test SSO(SSO 테스트) 페이지에서 모든 프록시 클러스터 노드를 통해 테스트 SSO를 시작하고 모든 것이 성공했는지 확인합니다. 이를 위해서는 프록시 노드를 리버스 하려면 클라이언트 시스템 연결이 필요합니다.

성능

nmon 툴로 만든 동급 최고의 성능 캡처에 대한 데이터 분석은 Finesse Release 12.6(1)ES2 소프트웨어 다운로드 페이지(**load_result.zip**)에서 확인할 수 있습니다. 데이터는 데스크톱 및 수퍼바이저 작업에 대한 프록시 상태를 나타내며, 샘플 2000 UCCE 구축에서 SSO 로그인을 사용하는 경우, 사용자 2000명을 위한 기본 레이아웃에 구성된 대로 8시간 동안 CUIC LD 보고서를 나타냅니다. 비교 가능한 하드웨어에서 Nginx를 사용하여 설치에 필요한 컴퓨팅, 디스크 및 네트워크 요구 사항을 도출하는 데 사용할 수 있습니다.

문제 해결

SELinux

1. Nginx가 기본적으로 시작되지 않거나 Finesse 에이전트 데스크톱에 액세스할 수 없는 경우 다음 명령을 사용하여 SELinux를 허용 모드로 설정합니다.
   ```
   setenforce 0
   ```
2. Nginx를 **systemctl restart nginx** 명령을 실행합니다.
3. 위반은 /var**/log/messages** 및 **/var/log/audit/audit.log에서** 사용할 수 있습니다.
4. 다음 명령 중 하나를 사용하여 이러한 위반을 해결할 수 있는 규칙을 사용하여 .te 파일을 재생성해야 합니다.
   ```
   cat /var/log/audit/audit.log | audit2allow -m nginx1 > nginx1.te. # this will create
   nginx1.te file
                   or
   ausearch -c 'nginx' --raw | audit2allow -M  my-nginx  # this will create my-nginx.te file
   ```
5. **selinux-nginx-rhel-master-modified/nginx** 디렉토리에 있는 원래 **nginx.te** 파일을 새로 생성된 허용 규칙으로 업데이트합니다.
6. 와 같은 방식으로 컴파일 **make** 명령을 실행합니다.
7. nginx.pp 파일이 다시 생성됩니다.
8. semodule 명령으로 정책을 로드합니다.
   ```
   semodule -i nginx.pp
   ```
9. SELinux에서 다음 명령을 사용하여 모드를 시행하도록 합니다.
   ```
   setenforce
   ```

10. 시스템을 재부팅합니다.
11. 필요한 위반이 수정될 때까지 이 절차를 반복합니다.