Red Hat Enterprise Linux 9

웹 서버 및 역방향 프록시 배포

Red Hat Enterprise Linux 9에 웹 서버 배포 및 역방향 프록시 배포 가이드

Last Updated: 2024-07-19

Red Hat Enterprise Linux 9 웹 서버 및 역방향 프록시 배포

Red Hat Enterprise Linux 9에 웹 서버 배포 및 역방향 프록시 배포 가이드

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

법적 공지

초록

이 문서에서는 Red Hat Enterprise Linux 9에서 웹 및 프록시 서버를 구성하고 실행하는 방법을 설명합니다: Apache HTTP 서버, NGINX 및 Squid.

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 CTO Chris Wright의 메시지 를 참조하십시오.

RED HAT 문서에 관한 피드백 제공

문서에 대한 피드백에 감사드립니다. 어떻게 개선할 수 있는지 알려주십시오.

특정 문구에 대한 의견 제출

1. **Multi-page HTML** 형식으로 설명서를 보고 페이지가 완전히 로드된 후 오른쪽 상단 모서리에 **피드백** 버튼이 표시되는지 확인합니다.
2. 커서를 사용하여 주석 처리할 텍스트 부분을 강조 표시합니다.
3. 강조 표시된 텍스트 옆에 표시되는 **피드백 추가** 버튼을 클릭합니다.
4. 의견을 추가하고 **제출** 을 클릭합니다.

Bugzilla를 통해 피드백 제출(등록 필요)

1. Bugzilla 웹 사이트에 로그인합니다.
2. **버전** 메뉴에서 올바른 버전을 선택합니다.
3. **Summary** (요약) 필드에 설명 제목을 입력합니다.
4. **Description** (설명) 필드에 개선을 위한 제안을 입력합니다. 문서의 관련 부분에 대한 링크를 포함합니다.
5. **버그 제출**을 클릭합니다.

1장. APACHE HTTP 웹 서버 설정

1.1. APACHE HTTP 웹 서버 개요

*웹 서버*는 웹을 통해 클라이언트에 콘텐츠를 제공하는 네트워크 서비스입니다. 이는 일반적으로 웹 페이지를 의미하지만 다른 모든 문서도 될 수 있습니다. 웹 서버는 *하이퍼텍스트 전송 프로토콜* (**HTTP**)을 사용하므로 HTTP 서버라고도 합니다.

Apache HTTP Server인 **httpd**는 Apache Software Foundation에서 개발한 오픈 소스 웹 서버입니다.

이전 Red Hat Enterprise Linux 릴리스에서 업그레이드하는 경우 그에 따라 **httpd** 서비스 구성을 업데이트해야 합니다. 이 섹션에서는 새로 추가된 일부 기능을 살펴보고 이전 구성 파일의 업데이트에 대해 설명합니다.

1.2. APACHE HTTP SERVER에서 주요 변경 사항

RHEL 9는 Apache HTTP Server 버전 2.4.48을 제공합니다. RHEL 8과 함께 배포된 2.4.37 버전에 비해 주요 변경 사항은 다음과 같습니다.

- Apache HTTP Server Control Interface (**apachectl**):
  - 이제 **apachectl 상태** 출력에 대해 **systemctl** pager가 비활성화되어 있습니다.
  - 이제 추가 인수를 전달하면 경고를 제공하는 대신 **apachectl** 명령이 실패합니다.
  - 이제 **apachectl normal-stop** 명령이 즉시 반환됩니다.
  - 이제 **apachectl configtest** 명령은 SELinux 컨텍스트를 변경하지 않고 **httpd -t** 명령을 실행합니다.
  - RHEL의 **apachectl(8)** 도움말 페이지는 이제 업스트림 **apachectl** 과 완전히 다른 문서를 사용합니다.
- Apache eXtenSion 툴(**apxs**):
  - **/usr/bin/apxs** 명령은 더 이상 **httpd** 패키지를 빌드할 때 컴파일러 최적화 플래그를 적용으로 사용하거나 노출하지 않습니다. 이제 **/usr/lib64/httpd/build/vendor-apxs** 명령을 사용하여 **httpd** 를 빌드하는 데 사용된 것과 동일한 컴파일러 플래그를 적용할 수 있습니다. **vendor-apxs** 명령을 사용하려면 먼저 **redhat-rpm-config** 패키지를 설치해야 합니다.
- Apache 모듈:
  - **mod_lua** 모듈이 이제 별도의 패키지로 제공됩니다.
- 구성 구문 변경 사항:
  - **mod_access_compat** 모듈에서 제공하는 더 이상 사용되지 않는 **Allow** 지시문에서 주석 (**#** 문자)이 이제 자동으로 무시되는 대신 구문 오류를 트리거합니다.
- 기타 변경 사항:
  - 이제 커널 스레드 ID가 오류 로그 메시지에서 직접 사용되어 정확하고 간결하게 만듭니다.
  - 많은 사소한 개선 사항 및 버그 수정
  - 모듈 작성자는 다양한 새 인터페이스를 사용할 수 있습니다.

RHEL 8 이후 **httpd** 모듈 API에 대한 이전 버전과 호환되지 않는 변경 사항은 없습니다.

Apache HTTP Server 2.4는 RPM 패키지로 쉽게 설치할 수 있는 이 Application Stream의 초기 버전입니다.

1.3. APACHE 구성 파일

httpd 는 기본적으로 시작 후 구성 파일을 읽습니다. 아래 표에서 구성 파일 위치 목록을 확인할 수 있습니다.

표 1.1. httpd 서비스 구성 파일

경로	설명
/etc/httpd/conf/httpd.conf	기본 구성 파일입니다.
/etc/httpd/conf.d/	기본 구성 파일에 포함된 구성 파일의 보조 디렉토리입니다.
/etc/httpd/conf.modules.d/	Red Hat Enterprise Linux에 패키징된 설치된 동적 모듈을 로드하는 구성 파일의 보조 디렉토리입니다. 기본 구성에서 이러한 구성 파일이 먼저 처리됩니다.

기본 구성은 대부분의 상황에 적합하지만 다른 구성 옵션도 사용할 수 있습니다. 변경 사항을 적용하려면 먼저 웹 서버를 다시 시작합니다.

구성에 가능한 오류가 있는지 확인하려면 쉘 프롬프트에서 다음을 입력합니다.

```
# apachectl configtest
Syntax OK
```

실수로부터 더 쉽게 복구하려면 편집하기 전에 원본 파일의 복사본을 만드십시오.

1.4. HTTPD 서비스 관리

이 섹션에서는 **httpd** 서비스를 시작, 중지 및 다시 시작하는 방법을 설명합니다.

사전 요구 사항

- Apache HTTP Server가 설치되어 있어야 합니다.

절차

- **httpd** 서비스를 시작하려면 다음을 입력합니다.

  ```
  # systemctl start httpd
  ```

- **httpd** 서비스를 중지하려면 다음을 입력합니다.

  ```
  # systemctl stop httpd
  ```

- **httpd** 서비스를 다시 시작하려면 다음을 입력합니다.

```
# systemctl restart httpd
```

1.5. 단일 인스턴스 APACHE HTTP SERVER 설정

이 섹션에서는 정적 HTML 콘텐츠를 제공하기 위해 단일 인스턴스 Apache HTTP Server를 설정하는 방법을 설명합니다.

웹 서버가 서버와 연결된 모든 도메인에 동일한 콘텐츠를 제공해야 하는 경우 이 섹션의 절차를 따릅니다. 도메인마다 다른 콘텐츠를 제공하려면 이름 기반 가상 호스트를 설정합니다. 자세한 내용은 Apache 이름 기반 가상 호스트 구성을 참조하십시오.

절차

1. **httpd** 패키지를 설치합니다.

```
# dnf install httpd
```

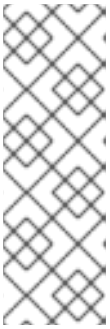
2. **firewalld** 를 사용하는 경우 로컬 방화벽에서 TCP 포트 **80** 을 엽니다.

```
# firewall-cmd --permanent --add-port=80/tcp
# firewall-cmd --reload
```

3. **httpd** 서비스를 활성화하고 시작합니다.

```
# systemctl enable --now httpd
```

4. 선택 사항: HTML 파일을 **/var/www/html/** 디렉터리에 추가합니다.



참고

콘텐츠를 **/var/www/html/**에 추가할 때 기본적으로 **httpd**가 실행되는 사용자가 파일 및 디렉터리를 읽을 수 있어야 합니다. 콘텐츠 소유자는 **root** 사용자 및 **root** 사용자 그룹 또는 다른 사용자 또는 관리자가 선택한 그룹일 수 있습니다. 콘텐츠 소유자가 **root** 사용자 및 **root** 사용자 그룹인 경우 다른 사용자가 파일을 읽을 수 있어야 합니다. 모든 파일 및 디렉터리에 대한 SELinux 컨텍스트는 기본적으로 **/var/www** 디렉터리 내의 모든 콘텐츠에 적용되는 **httpd_sys_content_t**이어야 합니다.

검증 단계

- 웹 브라우저와 함께 **http://*server_IP_or_host_name*/**에 연결합니다. **/var/www/html/** 디렉터리가 비어 있거나 **index.html** 또는 **index.htm** 파일이 포함되어 있지 않은 경우 Apache에서 **Red Hat Enterprise Linux Test Page**를 표시합니다. **/var/www/html/**에 다른 이름의 HTML 파일이 포함된 경우 **http://*server_IP_or_host_name*/*example.html***과 같은 URL을 해당 파일에 입력하여 로드할 수 있습니다.

추가 리소스

- Apache Manual: Apache HTTP 서버 설명서 설치 .
- **httpd.service(8)** 매뉴얼 페이지를 참조하십시오.

1.6. APACHE 이름 기반 가상 호스트 구성

이름 기반 가상 호스트를 사용하면 Apache가 서버의 IP 주소로 해석되는 다양한 도메인에 다양한 콘텐츠를 제공할 수 있습니다.

이 섹션의 절차에서는 별도의 문서 루트 디렉터리를 사용하여 **example.com** 및 **example.net** 도메인의 가상 호스트 설정에 대해 설명합니다. 두 가상 호스트는 모두 정적 HTML 콘텐츠를 제공합니다.

사전 요구 사항

- 클라이언트 및 웹 서버는 **example.com** 및 **example.net** 도메인을 웹 서버의 IP 주소로 확인합니다.
  이러한 항목을 DNS 서버에 수동으로 추가해야 합니다.

절차

1. **httpd** 패키지를 설치합니다.

```
# dnf install httpd
```

2. **/etc/httpd/conf/httpd.conf** 파일을 편집합니다.

   a. **example.com** 도메인에 다음 가상 호스트 구성을 추가합니다.

```
<VirtualHost *:80>
    DocumentRoot "/var/www/example.com/"
    ServerName example.com
    CustomLog /var/log/httpd/example.com_access.log combined
    ErrorLog /var/log/httpd/example.com_error.log
</VirtualHost>
```

   이러한 설정은 다음을 구성합니다.

   - **<VirtualHost *:80>** 지시문의 모든 설정은 이 가상 호스트에 따라 다릅니다.
   - **DocumentRoot**는 가상 호스트의 웹 콘텐츠 경로를 설정합니다.
   - **ServerName**은 이 가상 호스트가 콘텐츠를 제공하는 도메인을 설정합니다.
     여러 도메인을 설정하려면 **ServerAlias** 매개 변수를 구성에 추가하고 이 매개 변수에서 공백으로 구분된 추가 도메인을 지정합니다.
   - **CustomLog**는 가상 호스트의 액세스 로그 경로를 설정합니다.
   - **ErrorLog**는 가상 호스트의 오류 로그 경로를 설정합니다.

참고

Apache는 **ServerName** 및 **ServerAlias** 매개 변수에 설정된 도메인과 일치하지 않는 요청에도 구성에 있는 첫 번째 가상 호스트를 사용합니다. 여기에는 서버의 IP 주소로 전송된 요청도 포함됩니다.

3. **example.net** 도메인에 대해 유사한 가상 호스트 구성을 추가합니다.

```
<VirtualHost *:80>
```

```
    DocumentRoot "/var/www/example.net/"
    ServerName example.net
    CustomLog /var/log/httpd/example.net_access.log combined
    ErrorLog /var/log/httpd/example.net_error.log
</VirtualHost>
```

4. 두 가상 호스트 모두에 대한 문서 루트를 생성합니다.

```
# mkdir /var/www/example.com/
# mkdir /var/www/example.net/
```

5. **/var/www/** 내에 없는 **DocumentRoot** 매개변수에서 경로를 설정한 경우 두 문서 루트에서 **httpd_sys_content_t** 컨텍스트를 설정합니다.

```
# semanage fcontext -a -t httpd_sys_content_t "/srv/example.com(/.*)?"
# restorecon -Rv /srv/example.com/
# semanage fcontext -a -t httpd_sys_content_t "/srv/example.net(/.\*)?"
# restorecon -Rv /srv/example.net/
```

이러한 명령은 **/srv/example.com/** 및 **/srv/example.net/** 디렉터리에 **httpd_sys_content_t** 컨텍스트를 설정합니다.

restorecon 명령을 실행하려면 **policycoreutils-python-utils** 패키지를 설치해야 합니다.

6. **firewalld** 를 사용하는 경우 로컬 방화벽에서 포트 **80** 을 엽니다.

```
# firewall-cmd --permanent --add-port=80/tcp
# firewall-cmd --reload
```

7. **httpd** 서비스를 활성화하고 시작합니다.

```
# systemctl enable --now httpd
```

검증 단계

1. 각 가상 호스트의 문서 루트에 다른 예제 파일을 생성합니다.

```
# echo "vHost example.com" > /var/www/example.com/index.html
# echo "vHost example.net" > /var/www/example.net/index.html
```

2. 브라우저를 사용하고 **http://example.com**에 연결합니다. 웹 서버는 **example.com** 가상 호스트의 예제 파일을 보여줍니다.

3. 브라우저를 사용하고 **http://example.net**에 연결합니다. 웹 서버는 **example.net** 가상 호스트의 예제 파일을 보여줍니다.

추가 리소스

- Apache HTTP Server 설명서 설치 - 가상 호스트

1.7. APACHE HTTP 웹 서버에 대한 KERBEROS 인증 구성

Apache HTTP 웹 서버에서 Kerberos 인증을 수행하기 위해 RHEL 9에서는 **mod_auth_gssapi** Apache 모

듈을 사용합니다. **GSSAPI**(Generic Security Services API)는 Kerberos와 같은 보안 라이브러리 사용을 요청하는 애플리케이션의 인터페이스입니다. **gssproxy** 서비스를 사용하면 **httpd** 서버에 대해 권한 분리를 구현하여 보안 관점에서 이 프로세스를 최적화할 수 있습니다.

참고

mod_auth_gssapi 모듈은 제거된 **mod_auth_kerb** 모듈을 대체합니다.

사전 요구 사항

- **httpd**, **mod_auth_gssapi** 및 **gssproxy** 패키지가 설치됩니다.
- Apache 웹 서버가 설정되고 **httpd** 서비스가 실행 중입니다.

1.7.1. IdM 환경에서 GSS-Proxy 설정

다음 절차에서는 Apache HTTP 웹 서버에서 Kerberos 인증을 수행하기 위해 **GSS-Proxy**를 설정하는 방법을 설명합니다.

절차

1. 서비스 주체를 생성하여 HTTP/<SERVER_NAME>@realm 주체의 **keytab** 파일에 대한 액세스를 활성화합니다.

```
# ipa service-add HTTP/<SERVER_NAME>
```

2. **/etc/gssproxy/http.keytab** 파일에 저장된 **keytab** 탭을 검색합니다.

```
# ipa-getkeytab -s $(awk '/^server =/ {print $3}' /etc/ipa/default.conf) -k
/etc/gssproxy/http.keytab -p HTTP/$(hostname -f)
```

   이 단계에서는 권한을 400으로 설정하면 **root** 사용자만 **keytab** 파일에 액세스할 수 있습니다. **apache** 사용자는 그렇지 않습니다.

3. 다음 콘텐츠를 사용하여 **/etc/gssproxy/80-httpd.conf** 파일을 만듭니다.

```
[service/HTTP]
  mechs = krb5
  cred_store = keytab:/etc/gssproxy/http.keytab
  cred_store = ccache:/var/lib/gssproxy/clients/krb5cc_%U
  euid = apache
```

4. **gssproxy** 서비스를 재시작하고 활성화합니다.

```
# systemctl restart gssproxy.service
# systemctl enable gssproxy.service
```

추가 리소스

- **gssproxy(8)** 매뉴얼 페이지
- **gssproxy-mech(8)** 매뉴얼 페이지

- **gssproxy.conf(5)** 매뉴얼 페이지

1.7.2. Apache HTTP 웹 서버에서 공유하는 디렉터리에 대한 Kerberos 인증 구성

다음 절차에서는 **/var/www/html/private/** 디렉터리에 대한 Kerberos 인증을 구성하는 방법을 설명합니다.

사전 요구 사항

- **gssproxy** 서비스가 구성되어 실행 중입니다.

절차

1. **/var/www/html/private/** 디렉터리를 보호하도록 **mod_auth_gssapi** 모듈을 구성합니다.

```
<Location /var/www/html/private>
  AuthType GSSAPI
  AuthName "GSSAPI Login"
  Require valid-user
</Location>
```

2. 다음 콘텐츠를 사용하여 **/etc/systemd/system/httpd.service** 파일을 만듭니다.

```
.include /lib/systemd/system/httpd.service
[Service]
Environment=GSS_USE_PROXY=1
```

3. **systemd** 구성을 다시 로드합니다.

```
# systemctl daemon-reload
```

4. **httpd** 서비스를 다시 시작합니다.

```
# systemctl restart httpd.service
```

검증 단계

1. Kerberos 티켓을 받습니다.

```
# kinit
```

2. 브라우저에서 보호된 디렉터리의 URL을 엽니다.

1.8. APACHE HTTP SERVER에서 TLS 암호화 구성

기본적으로 Apache는 암호화되지 않은 HTTP 연결을 사용하여 클라이언트에 콘텐츠를 제공합니다. 이 섹션에서는 TLS 암호화를 활성화하고 Apache HTTP Server에서 자주 사용하는 암호화 관련 설정을 구성하는 방법에 대해 설명합니다.

사전 요구 사항

- Apache HTTP Server가 설치되어 실행 중입니다.

1.8.1. Apache HTTP Server에 TLS 암호화 추가

이 섹션에서는 **example.com** 도메인의 Apache HTTP Server에서 TLS 암호화를 활성화하는 방법을 설명합니다.

사전 요구 사항

- Apache HTTP Server가 설치되어 실행 중입니다.
- 개인 키는 **/etc/pki/tls/private/example.com.key** 파일에 저장됩니다.
  개인 키 및 CSR(인증서 서명 요청) 생성 및 CA(인증 기관)에서 인증서를 요청하는 방법에 대한 자세한 내용은 CA 문서를 참조하십시오. 또는 CA에서 ACME 프로토콜을 지원하는 경우 **mod_md** 모듈을 사용하여 TLS 인증서 검색 및 프로비저닝을 자동화할 수 있습니다.
- TLS 인증서는 **/etc/pki/tls/certs/example.com.crt** 파일에 저장됩니다. 다른 경로를 사용하는 경우 절차의 해당 단계를 조정합니다.
- CA 인증서는 **/etc/pki/tls/certs/ca.crt** 파일에 저장됩니다. 다른 경로를 사용하는 경우 절차의 해당 단계를 조정합니다.
- 클라이언트 및 웹 서버는 서버의 호스트 이름을 웹 서버의 IP 주소로 확인합니다.

절차

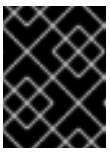
1. **mod_ssl** 패키지를 설치합니다.

```
# dnf install mod_ssl
```

2. **/etc/httpd/conf.d/ssl.conf** 파일을 편집하고 다음 설정을 **<VirtualHost _default_:443>** 지시문에 추가합니다.

   a. 서버 이름을 설정합니다.

```
ServerName example.com
```



중요

서버 이름은 인증서의 **Common Name** 필드에 설정된 항목과 일치해야 합니다.

   b. 선택 사항: 인증서에 SAN(**Subject Alt Names**) 필드에 추가 호스트 이름이 포함된 경우 이러한 호스트 이름에도 TLS 암호화를 제공하도록 **mod_ssl**을 구성할 수 있습니다. 이를 구성하려면 해당 이름으로 **ServerAliases** 매개변수를 추가합니다.

```
ServerAlias www.example.com server.example.com
```

   c. 개인 키, 서버 인증서 및 CA 인증서에 대한 경로를 설정합니다.

```
SSLCertificateKeyFile "/etc/pki/tls/private/example.com.key"
SSLCertificateFile "/etc/pki/tls/certs/example.com.crt"
SSLCACertificateFile "/etc/pki/tls/certs/ca.crt"
```

3. 보안상의 이유로 **root** 사용자만 개인 키 파일에 액세스할 수 있도록 구성합니다.

```
# chown root:root /etc/pki/tls/private/example.com.key
# chmod 600 /etc/pki/tls/private/example.com.key
```

주의

권한이 없는 사용자가 개인 키에 액세스한 경우 인증서를 취소하고 새 개인 키를 만들고 새 인증서를 요청합니다. 그렇지 않으면 TLS 연결이 더 이상 안전하지 않습니다.

4. **firewalld** 를 사용하는 경우 로컬 방화벽에서 포트 **443** 을 엽니다.

```
# firewall-cmd --permanent --add-port=443/tcp
# firewall-cmd --reload
```

5. **httpd** 서비스를 다시 시작합니다.

```
# systemctl restart httpd
```

참고

개인 키 파일을 암호로 보호한 경우 **httpd** 서비스가 시작될 때마다 이 암호를 입력해야 합니다.

검증 단계

- 브라우저를 사용하고 **https://*example.com***에 연결합니다.

추가 리소스

- SSL/TLS 암호화
- RHEL 9의 TLS에 대한 보안 고려 사항

1.8.2. Apache HTTP Server에서 지원되는 TLS 프로토콜 버전 설정

기본적으로 RHEL의 Apache HTTP Server는 최신 브라우저와 호환되는 안전한 기본값을 정의하는 시스템 전체 암호화 정책을 사용합니다. 예를 들어 **DEFAULT** 정책은 apache에서 **TLSv1.2** 및 **TLSv1.3** 프로토콜 버전만 사용하도록 정의합니다.

이 섹션에서는 Apache HTTP Server에서 지원하는 TLS 프로토콜 버전을 수동으로 구성하는 방법을 설명합니다. 환경에서 특정 TLS 프로토콜 버전만 활성화해야 하는 경우 절차를 따르십시오. 예를 들면 다음과 같습니다.

- 환경에 해당 클라이언트가 클라이언트가 취약한 **TLS1** (TLSv1.0) 또는 **TLS1.1** 프로토콜을 사용할 수도 있어야 하는 경우
- Apache가 **TLSv1.2** 또는 **TLSv1.3** 프로토콜만 지원하도록 구성하려는 경우

사전 요구 사항

- Apache HTTP 서버에 TLS 암호화 추가에 설명된 대로 서버에서 TLS 암호화가 활성화되어 있습니다.

절차

1. **/etc/httpd/conf/httpd.conf** 파일을 편집하고 다음 설정을 TLS 프로토콜 버전을 설정하려는 **<VirtualHost>** 지시문에 추가합니다. 예를 들어 **TLSv1.3** 프로토콜만 활성화하려면 다음을 수행합니다.

```
SSLProtocol -All TLSv1.3
```

2. **httpd** 서비스를 다시 시작합니다.

```
# systemctl restart httpd
```

검증 단계

1. 다음 명령을 사용하여 서버가 **TLSv1.3**을 지원하는지 확인합니다:

```
# openssl s_client -connect example.com:443 -tls1_3
```

2. 다음 명령을 사용하여 서버가 **TLSv1.2**를 지원하지 않는지 확인합니다.

```
# openssl s_client -connect example.com:443 -tls1_2
```

서버가 프로토콜을 지원하지 않으면 명령에서 오류를 반환합니다.

```
140111600609088:error:1409442E:SSL routines:ssl3_read_bytes:tlsv1 alert protocol
version:ssl/record/rec_layer_s3.c:1543:SSL alert number 70
```

3. 선택 사항: 다른 TLS 프로토콜 버전에 대해 명령을 반복합니다.

추가 리소스

- **update-crypto-policies(8)** 도움말 페이지
- 시스템 전체 암호화 정책 사용.
- **SSLProtocol** 매개변수에 대한 자세한 내용은 Apache 설명서의 **mod_ssl** 설명서를 참조하십시오. Apache HTTP 서버 설명서 설치.

1.8.3. Apache HTTP Server에서 지원되는 암호 설정

기본적으로 Apache HTTP 서버는 최근 브라우저와 호환되는 안전한 기본값을 정의하는 시스템 전체 암호화 정책을 사용합니다. 시스템 전체에서 허용하는 암호 목록은 **/etc/crypto-policies/back-ends/openssl.config** 파일을 참조하십시오.

이 섹션에서는 Apache HTTP Server에서 지원하는 암호를 수동으로 구성하는 방법을 설명합니다. 환경에 특정 암호가 필요한 경우 절차를 따르십시오.

사전 요구 사항

- Apache HTTP 서버에 TLS 암호화 추가에 설명된 대로 서버에서 TLS 암호화가 활성화되어 있습니다.

절차

1. **/etc/httpd/conf/httpd.conf** 파일을 편집하고 **SSLCipherSuite** 매개변수를 TLS 암호를 설정하려는 **<VirtualHost>** 지시문에 추가합니다.

```
SSLCipherSuite
"EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH:!SHA1:!SHA256"
```

   이 예제에서는 **EECDH+AESGCM,EDH+AESGCM,AES256+EECDH** 및 **AES256+EDH** 암호만 활성화하며 **SHA1** 및 **SHA256** 메시지 인증 코드(MAC)를 사용하는 모든 암호를 비활성화합니다.

2. **httpd** 서비스를 다시 시작합니다.

```
# systemctl restart httpd
```

검증 단계

1. Apache HTTP Server에서 지원하는 암호 목록을 표시하려면 다음을 수행합니다.

   a. **nmap** 패키지를 설치합니다.

```
# dnf install nmap
```

   b. **nmap** 유틸리티를 사용하여 지원되는 암호를 표시합니다.

```
# nmap --script ssl-enum-ciphers -p 443 example.com
...
PORT    STATE SERVICE
443/tcp open  https
| ssl-enum-ciphers:
|   TLSv1.2:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (ecdh_x25519) - A
|       TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (dh 2048) - A
|       TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (ecdh_x25519) - A
...
```

추가 리소스

- **update-crypto-policies(8)** 도움말 페이지
- 시스템 전체 암호화 정책 사용 .
- SSLCipherSuite

1.9. TLS 클라이언트 인증서 인증 구성

관리자는 클라이언트 인증서 인증을 통해 인증서를 사용하여 인증한 사용자만 웹 서버의 리소스에 액세스할 수 있도록 허용할 수 있습니다. 이 섹션에서는 **/var/www/html/Example/** 디렉터리에 대한 클라이언트 인증서 인증을 구성하는 방법을 설명합니다.

Apache HTTP Server에서 TLS 1.3 프로토콜을 사용하는 경우 특정 클라이언트에 추가 구성이 필요합니다. 예를 들어 Firefox에서 **about:config** 메뉴의 **security.tls.enable_post_handshake_auth** 매개 변수를 **true**로 설정합니다. 자세한 내용은 Red Hat Enterprise Linux 8의 전송 계층 보안 버전 1.3 을 참조하십시오.

사전 요구 사항

- Apache HTTP 서버에 TLS 암호화 추가에 설명된 대로 서버에서 TLS 암호화가 활성화되어 있습니다.

절차

1. **/etc/httpd/conf/httpd.conf** 파일을 편집하고 다음 설정을 클라이언트 인증을 구성하려는 **<VirtualHost>** 지시문에 추가합니다.

```
<Directory "/var/www/html/Example/">
  SSLVerifyClient require
</Directory>
```

   SSLverifyClient require를 설정해야 클라이언트가 **/var/www/html/Example/** 디렉터리의 콘텐츠에 액세스하기 전에 서버가 클라이언트 인증서의 유효성을 검사해야 합니다.

2. **httpd** 서비스를 다시 시작합니다.

```
# systemctl restart httpd
```

검증 단계

1. **curl** 유틸리티를 사용하여 클라이언트 인증 없이 **https://example.com/Example/** URL에 액세스합니다.

```
$ curl https://example.com/Example/
curl: (56) OpenSSL SSL_read: error:1409445C:SSL routines:ssl3_read_bytes:tlsv13 **alert certificate required**, errno 0
```

   이 오류는 웹 서버에 클라이언트 인증서 인증이 필요함을 나타냅니다.

2. 클라이언트 개인 키 및 인증서와 CA 인증서를 **curl**에 전달하여 클라이언트 인증으로 동일한 URL에 액세스합니다.

```
$ curl --cacert ca.crt --key client.key --cert client.crt https://example.com/Example/
```

   요청이 성공하면 **curl**은 **/var/www/html/Example/** 디렉터리에 저장된 **index.html** 파일을 표시합니다.

추가 리소스

- mod_ssl 구성

1.10. MODSECURITY을 사용하여 웹 서버에서 웹 애플리케이션 보안

ModSecurity는 Apache, Nginx 및 IIS와 같은 다양한 웹 서버에서 지원하는 오픈 소스 웹 애플리케이션 방화벽입니다. ModSecurity는 서버 구성을 위해 사용자 정의 가능한 규칙 세트를 제공합니다.

mod_security-crs 패키지에는 웹 사이트 간 스크립팅, 잘못된 사용자 에이전트, SQL 인젝션, 트로이 목마, 세션 hijacking 및 기타 악용에 대한 규칙이 있는 핵심 규칙 세트(CRS)가 포함되어 있습니다.

1.10.1. Apache용 ModSecurity 웹 기반 애플리케이션 방화벽 배포

ModSecurity을 배포하여 웹 서버에서 웹 기반 애플리케이션을 실행하는 것과 관련된 위험을 줄이려면 Apache HTTP 서버에 대한 **mod_security** 및 **mod_security_crs** 패키지를 설치하십시오. **mod_security_crs** 패키지는 ModSecurity 웹 기반 애플리케이션 방화벽(WAF) 모듈에 대한 핵심 규칙 세트(CRS)를 제공합니다.

절차

1. **mod_security**,**mod_security_crs**, **httpd** 패키지를 설치합니다.

```
# dnf install -y mod_security mod_security_crs httpd
```

2. **httpd** 서버를 시작합니다.

```
# systemctl restart httpd
```

검증

1. Apache HTTP 서버에서 ModSecurity 웹 기반 애플리케이션 방화벽이 활성화되어 있는지 확인합니다.

```
# httpd -M | grep security
 security2_module (shared)
```

2. **/etc/httpd/modsecurity.d/activated_rules/** 디렉터리에 **mod_security_crs** 에서 제공하는 규칙이 포함되어 있는지 확인합니다.

```
# ls /etc/httpd/modsecurity.d/activated_rules/
...
REQUEST-921-PROTOCOL-ATTACK.conf
REQUEST-930-APPLICATION-ATTACK-LFI.conf
...
```

추가 리소스

- Red Hat JBoss Core Services ModSecurity Guide
- Linux sysadmins용 웹 애플리케이션 방화벽 소개

1.10.2. ModSecurity에 사용자 정의 규칙 추가

ModSecurity 코어 규칙 세트(CRS)에 포함된 규칙이 시나리오에 맞지 않고 가능한 추가 공격을 방지하려면 사용자 지정 규칙을 ModSecurity 웹 기반 애플리케이션 방화벽에서 사용하는 규칙 세트에 추가할 수 있습니다. 다음 예제에서는 간단한 규칙을 추가하는 방법을 보여줍니다. 더 복잡한 규칙을 만들려면 ModSecurity Wiki 웹 사이트의 참조 설명서를 참조하십시오.

사전 요구 사항

- Apache용 ModSecurity가 설치 및 활성화되어 있습니다.

절차

1. 선택한 텍스트 편집기에서 **/etc/httpd/conf.d/mod_security.conf** 파일을 엽니다. 예를 들면 다음과 같습니다.

```
# vi /etc/httpd/conf.d/mod_security.conf
```

2. **SecRuleEngine On**:로 시작하는 행 다음에 다음 예제 규칙을 추가합니다.

```
SecRule ARGS:data "@contains evil" "deny,status:403,msg:'param data contains evil data',id:1"
```

이전 규칙은 **data** 매개 변수에 잘못된 문자열이 포함된 경우 사용자에게 리소스 사용 **을 금지** 합니다.

3. 변경 사항을 저장하고 편집기를 종료합니다.

4. **httpd** 서버를 다시 시작합니다.

```
# systemctl restart httpd
```

검증

1. ***테스트*.html** 페이지를 생성합니다.

```
# echo "mod_security test" > /var/www/html/test.html
```

2. **httpd** 서버를 다시 시작합니다.

```
# systemctl restart httpd
```

3. HTTP 요청의 **GET** 변수에 악성 데이터 없이 ***test.html*** 을 요청합니다.

```
$ curl http://localhost/test.html?data=good

mod_security test
```

4. HTTP 요청의 **GET** 변수에 악의적인 데이터를 사용하여 ***test.html*** 을 요청합니다.

```
$ curl localhost/test.html?data=xxxevilxxx

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>
```

5. **/var/log/httpd/error_log** 파일을 확인하고 악한 데이터 메시지가 **포함된 param 데이터로 액세스를 거부하는** 것에 대한 로그 항목을 찾습니다.

```
[Wed May 25 08:01:31.036297 2022] [:error] [pid 5839:tid 139874434791168] [client
```

```
::1:45658] [client ::1] ModSecurity: Access denied with code 403 (phase 2). String match
"evil" at ARGS:data. [file "/etc/httpd/conf.d/mod_security.conf"] [line "4"] [id "1"] [msg "param
data contains evil data"] [hostname "localhost"] [uri "/test.html"] [unique_id
"Yo4amwIdsBG3yZqSzh2GuwAAAIY"]
```

추가 리소스

- ModSecurity Wiki

1.11. APACHE HTTP SERVER 수동 설치

이 섹션에서는 Apache HTTP Server 설명서를 설치하는 방법을 설명합니다. 이 설명서는 다음과 같은 자세한 문서를 제공합니다. 예를 들면 다음과 같습니다.

- 구성 매개변수 및 지시문
- 성능 튜닝
- 인증 설정
- 모듈
- 컨텐츠 캐싱
- 보안 팁
- TLS 암호화 구성

설명서를 설치한 후 웹 브라우저를 사용하여 표시할 수 있습니다.

사전 요구 사항

- Apache HTTP Server가 설치되어 실행 중입니다.

절차

1. **httpd-manual** 패키지를 설치합니다.

```
# dnf install httpd-manual
```

2. 선택 사항: 기본적으로 Apache HTTP 서버에 연결하는 모든 클라이언트는 설명서를 표시할 수 있습니다. **192.0.2.0/24** 서브넷과 같은 특정 IP 범위에 대한 액세스를 제한하려면 **/etc/httpd/conf.d/manual.conf** 파일을 편집하고 **Require ip 192.0.2.0/24** 설정을 **<Directory "/usr/share/httpd/manual">** 지시문에 추가합니다.

```
<Directory "/usr/share/httpd/manual">
...
   **Require ip 192.0.2.0/24**
...
</Directory>
```

3. **httpd** 서비스를 다시 시작합니다.

```
# systemctl restart httpd
```

검증 단계

1. Apache HTTP Server 설명서를 표시하려면 웹 브라우저로 **http://*host_name_or_IP_address*/manual/**에 연결합니다.

1.12. APACHE 모듈 작업

httpd 서비스는 모듈식 애플리케이션이며 다양한 **DSO**(*Dynamic Shared Objects*)로 확장할 수 있습니다. *동적 공유 오브젝트* 는 필요에 따라 런타임에 동적으로 로드 또는 언로드할 수 있는 모듈입니다. 이러한 모듈은 **/usr/lib64/httpd/modules/** 디렉토리에서 찾을 수 있습니다.

1.12.1. DSO 모듈 로드

관리자는 서버를 로드할 모듈을 구성하여 서버에 포함할 기능을 선택할 수 있습니다. 특정 DSO 모듈을 로드하려면 **LoadModule** 지시문을 사용합니다. 별도의 패키지에서 제공하는 모듈에는 **/etc/httpd/conf.modules.d/** 디렉터리에 고유한 구성 파일이 있는 경우가 많습니다.

사전 요구 사항

- **httpd** 패키지가 설치되어 있습니다.

절차

1. **/etc/httpd/conf.modules.d/** 디렉터리의 구성 파일에서 모듈 이름을 검색합니다.

```
# grep mod_ssl.so /etc/httpd/conf.modules.d/*
```

2. 모듈 이름을 찾은 구성 파일을 편집하고 모듈의 **LoadModule** 지시문의 주석을 해제합니다.

```
LoadModule ssl_module modules/mod_ssl.so
```

3. 예를 들어 RHEL 패키지에서 모듈을 제공하지 않기 때문에 모듈이 없는 경우 다음 지시문을 사용하여 **/etc/httpd/conf.modules.d/30-example.conf** 와 같은 구성 파일을 만듭니다.

```
LoadModule ssl_module modules/<custom_module>.so
```

4. **httpd** 서비스를 다시 시작합니다.

```
# systemctl restart httpd
```

1.12.2. 사용자 정의 Apache 모듈 컴파일

고유한 모듈을 생성하고 **httpd-devel** 패키지의 도움을 받아 빌드할 수 있습니다. 여기에는 포함 파일, 헤더 파일, 모듈을 컴파일하는 데 필요한 **APache eXtenSion** (**apxs**) 유틸리티가 포함됩니다.

사전 요구 사항

- **httpd-devel** 패키지가 설치되어 있어야 합니다.

절차

- 다음 명령을 사용하여 사용자 정의 모듈을 빌드합니다.

-

```
# apxs -i -a -c module_name.c
```

검증 단계

- DSO 모듈 로드에 설명된 것과 동일한 방식으로 모듈을 로드 합니다.

1.13. APACHE 웹 서버 구성에서 사용하기 위해 NSS 데이터베이스에서 개인 키 및 인증서 내보내기

RHEL 8에서는 더 이상 Apache 웹 서버에 **mod_nss** 모듈을 제공하지 않으며, Red Hat은 **mod_ssl** 모듈을 사용하는 것이 좋습니다. 예를 들어 웹 서버를 RHEL 7에서 RHEL 8로 마이그레이션했기 때문에 개인 키와 인증서를 NSS(Network Security Services) 데이터베이스에 저장하는 경우 다음 절차에 따라 PEM(개인 정보 보호 보안 메일) 형식으로 키와 인증서를 추출합니다. 그런 다음 Apache HTTP 서버에서 TLS 암호화 구성에 설명된 대로 **mod_ssl** 구성에서 파일을 사용할 수 있습니다.

이 절차에서는 NSS 데이터베이스가 **/etc/httpd/alias/**에 저장되고 내보낸 개인 키와 인증서를 **/etc/pki/tls/** 디렉터리에 저장하는 것으로 가정합니다.

사전 요구 사항

- 개인 키, 인증서 및 CA(인증 기관) 인증서는 NSS 데이터베이스에 저장됩니다.

절차

1. NSS 데이터베이스의 인증서를 나열합니다.

```
# certutil -d /etc/httpd/alias/ -L
Certificate Nickname          Trust Attributes
                              SSL,S/MIME,JAR/XPI

Example CA                    C,,
Example Server Certificate    u,u,u
```

   다음 단계에서 인증서의 닉네임이 필요합니다.

2. 개인 키를 추출하려면 일시적으로 PKCS #12 파일로 키를 내보내야 합니다.

   a. 개인 키와 연결된 인증서의 닉네임을 사용하여 PKCS #12 파일로 키를 내보냅니다.

```
# pk12util -o /etc/pki/tls/private/export.p12 -d /etc/httpd/alias/ -n "Example Server Certificate"
Enter password for PKCS12 file: password
Re-enter password: password
pk12util: PKCS12 EXPORT SUCCESSFUL
```

   PKCS #12 파일에 암호를 설정해야 합니다. 다음 단계에서 이 암호가 필요합니다.

   b. PKCS #12 파일에서 개인 키를 내보냅니다.

```
# openssl pkcs12 -in /etc/pki/tls/private/export.p12 -out /etc/pki/tls/private/server.key -nocerts -nodes
Enter Import Password: password
MAC verified OK
```

c. 임시 PKCS #12 파일을 삭제합니다.

```
# rm /etc/pki/tls/private/export.p12
```

3. **root** 사용자만 이 파일에 액세스할 수 있도록 **/etc/pki/tls/private/server.key**에 대한 권한을 설정합니다.

```
# chown root:root /etc/pki/tls/private/server.key
# chmod 0600 /etc/pki/tls/private/server.key
```

4. NSS 데이터베이스에서 서버 인증서의 닉네임을 사용하여 CA 인증서를 내보냅니다.

```
# certutil -d /etc/httpd/alias/ -L -n "Example Server Certificate" -a -o
/etc/pki/tls/certs/server.crt
```

5. **root** 사용자만 이 파일에 액세스할 수 있도록 **/etc/pki/tls/certs/server.crt**에 대한 권한을 설정합니다.

```
# chown root:root /etc/pki/tls/certs/server.crt
# chmod 0600 /etc/pki/tls/certs/server.crt
```

6. NSS 데이터베이스에서 CA 인증서의 닉네임을 사용하여 CA 인증서를 내보냅니다.

```
# certutil -d /etc/httpd/alias/ -L -n "Example CA" -a -o /etc/pki/tls/certs/ca.crt
```

7. Apache HTTP 서버에서 TLS 암호화 구성 을 따라 Apache 웹 서버를 구성합니다.
   - **SSLCertificateKeyFile** 매개변수를 **/etc/pki/tls/private/server.key**로 설정합니다.
   - **SSLCertificateFile** 매개변수를 **/etc/pki/tls/certs/server.crt**로 설정합니다.
   - **SSLCACertificateFile** 매개변수를 **/etc/pki/tls/certs/ca.crt**로 설정합니다.

추가 리소스

- **certutil(1)** 매뉴얼 페이지
- **pk12util(1)** 매뉴얼 페이지
- **pkcs12(1ssl)** 매뉴얼 페이지

1.14. 추가 리소스

- **httpd(8)** 매뉴얼 페이지
- **httpd.service(8)** 매뉴얼 페이지
- **httpd.conf(5)** 매뉴얼 페이지
- **apachectl(8)** 매뉴얼 페이지
- Apache HTTP 서버의 Kerberos 인증: Apache httpd 작업에 GSS-Proxy 사용. Kerberos 사용은 Apache HTTP 서버에서 클라이언트 권한을 적용하는 대체 방법입니다.

- PKCS #11을 통해 암호화 하드웨어를 사용하도록 애플리케이션 구성 .

2장. NGINX 설정 및 구성

NGINX는 다음과 같이 사용할 수 있는 고성능 모듈식 서버입니다.

- 웹 서버
- 역방향 프록시
- 로드 밸런서

이 섹션에서는 이러한 시나리오에서 NGINX를 수행하는 방법을 설명합니다.

2.1. NGINX 설치 및 준비

Red Hat은 애플리케이션 스트림을 사용하여 다양한 버전의 NGINX를 제공합니다. 이 섹션에서는 다음을 수행하는 방법에 대해 설명합니다.

- 스트림 선택 및 NGINX를 설치
- 방화벽에서 필요한 포트를 열기
- **nginx** 서비스 활성화 및 시작

기본 구성을 사용하여 NGINX는 포트 **80**의 웹 서버로 실행되며 **/usr/share/nginx/html/** 디렉터리에서 콘텐츠를 제공합니다.

사전 요구 사항

- RHEL 9가 설치되어 있어야 합니다.
- 호스트가 Red Hat 고객 포털에 가입되어 있습니다.
- **firewalld** 서비스가 활성화 및 시작됩니다.

절차

1. 사용 가능한 NGINX 모듈 스트림을 표시합니다.

```
# yum module list nginx
Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)
Name       Stream       Profiles       Summary
nginx      1.14 [d]     common [d]     nginx webserver
nginx      1.16         common [d]     nginx webserver
...

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

2. 기본값과 다른 스트림을 설치하려면 스트림을 선택합니다.

```
# yum module enable nginx:stream_version
```

3. **nginx** 패키지를 설치합니다.

```
# yum install nginx
```

4. NGINX가 방화벽에 서비스를 제공해야 하는 포트를 엽니다. 예를 들어 **firewalld**에서 HTTP(포트 80) 및 HTTPS(포트 443)의 기본 포트를 열려면 다음을 입력합니다.

```
# firewall-cmd --permanent --add-port={80/tcp,443/tcp}
# firewall-cmd --reload
```

5. 시스템이 부팅될 때 **nginx** 서비스가 자동으로 시작되도록 활성화합니다.

```
# systemctl enable nginx
```

6. 선택적으로 **nginx** 서비스를 시작합니다.

```
# systemctl start nginx
```

기본 구성을 사용하지 않으려면 서비스를 시작하기 전에 이 단계를 건너뛰고 그에 따라 NGINX를 구성합니다.

검증 단계

1. **yum** 유틸리티를 사용하여 **nginx** 패키지가 설치되었는지 확인합니다.

```
# yum list installed nginx
Installed Packages
nginx.x86_64    1:1.14.1-9.module+el8.0.0+4108+af250afe    @rhel-8-for-x86_64-appstream-rpms
```

추가 리소스

- 서브스크립션 관리자 사용 및 구성
- 네트워크 보안

2.2. 다른 도메인에 다른 콘텐츠를 제공하는 웹 서버로 NGINX 구성

기본적으로 NGINX는 서버의 IP 주소와 연결된 모든 도메인 이름에 대해 클라이언트에 동일한 콘텐츠를 제공하는 웹 서버 역할을 합니다. 다음 절차에서는 NGINX를 구성하는 방법을 설명합니다.

- **/var/www/example.com/** 디렉터리의 콘텐츠를 사용하여 **example.com**도메인에 요청을 처리하는 방법
- **/var/www/example .net/ 디렉터리의 콘텐츠를 사용하여 example.net 도메인에 요청을 제공하려면 다음을 수행합니다.**
- 예를 들어, 서버의 IP 주소 또는 서버의 IP 주소와 연결된 다른 도메인에 대한 다른 모든 요청을 **/usr/share/nginx/html/** 디렉터리의 콘텐츠로 제공하려면 다음을 수행합니다.

사전 요구 사항

- NGINX 설치
- 클라이언트 및 웹 서버는 **example.com** 및 **example.net** 도메인을 웹 서버의 IP 주소로 확인합니다.
  이러한 항목을 DNS 서버에 수동으로 추가해야 합니다.

절차

1. **/etc/nginx/nginx.conf** 파일을 편집합니다.

   a. 기본적으로 **/etc/nginx/nginx.conf** 파일에는 catch-all 구성이 이미 포함되어 있습니다. 구성에서 이 부분을 삭제한 경우 **/etc/nginx/nginx.conf 파일의 http 블록에 다음 server 블록을 다시 추가합니다.**

   ```
   server {
       listen       80 default_server;
       listen       [::]:80 default_server;
       server_name  _;
       root         /usr/share/nginx/html;
   }
   ```

   이러한 설정은 다음을 구성합니다.

   - **listen** 지시문은 서비스에서 수신 대기하는 IP 주소와 포트를 정의합니다. 이 경우 NGINX는 모든 IPv4 및 IPv6 주소의 포트 **80** 에서 수신 대기합니다. **default_server** 매개 변수는 NGINX가 이 **서버** 블록을 IP 주소 및 포트와 일치하는 요청의 기본값으로 사용함을 나타냅니다.
   - **server_name** 매개 변수는 이 **서버** 블록을 담당하는 호스트 이름을 정의합니다. **server_name** 을 **_** 로 설정하여 이 **서버** 블록의 모든 호스트 이름을 수락하도록 NGINX를 구성합니다.
   - **root** 지시문은 이 **서버** 블록의 웹 콘텐츠 경로를 설정합니다.

   b. **example.com** 도메인에 대한 유사한 **server** 블록을 **http 블록에** 추가합니다.

   ```
   server {
       server_name  example.com;
       root         /var/www/example.com/;
       access_log   /var/log/nginx/example.com/access.log;
       error_log    /var/log/nginx/example.com/error.log;
   }
   ```

   - **access_log** 지시문은 이 도메인에 대한 별도의 액세스 로그 파일을 정의합니다.
   - **error_log** 지시문은 이 도메인에 대한 별도의 오류 로그 파일을 정의합니다.

   c. **example.net** 도메인에 대한 유사한 **server** 블록을 **http 블록에** 추가합니다.

   ```
   server {
       server_name  example.net;
       root         /var/www/example.net/;
       access_log   /var/log/nginx/example.net/access.log;
       error_log    /var/log/nginx/example.net/error.log;
   }
   ```

2. 두 도메인의 루트 디렉터리를 생성합니다.

   ```
   # mkdir -p /var/www/example.com/
   # mkdir -p /var/www/example.net/
   ```

3. 두 루트 디렉터리에 **httpd_sys_content_t** 컨텍스트를 설정합니다.

```
# semanage fcontext -a -t httpd_sys_content_t "/var/www/example.com(/.*)?"
# restorecon -Rv /var/www/example.com/
# semanage fcontext -a -t httpd_sys_content_t "/var/www/example.net(/.\*)?"
# restorecon -Rv /var/www/example.net/
```

이러한 명령은 /var/www/ **example.com/ 및 /var/www/example.net/** 디렉터리에 **httpd_sys_content_t** 컨텍스트를 설정합니다.

restorecon 명령을 실행하려면 **policycoreutils-python-utils** 패키지를 설치해야 합니다.

4. 두 도메인의 로그 디렉터리를 생성합니다.

```
# mkdir /var/log/nginx/example.com/
# mkdir /var/log/nginx/example.net/
```

5. **nginx** 서비스를 다시 시작합니다.

```
# systemctl restart nginx
```

검증 단계

1. 각 가상 호스트의 문서 루트에 다른 예제 파일을 생성합니다.

```
# echo "Content for example.com" > /var/www/example.com/index.html
# echo "Content for example.net" > /var/www/example.net/index.html
# echo "Catch All content" > /usr/share/nginx/html/index.html
```

2. 브라우저를 사용하고 **http://example.com**에 연결합니다. 웹 서버는 **/var/www/example.com/index.html** 파일의 예제 콘텐츠를 표시합니다.

3. 브라우저를 사용하고 **http://example.net**에 연결합니다. 웹 서버는 **/var/www/example.net/index.html** 파일의 예제 콘텐츠를 표시합니다.

4. 브라우저를 사용하고 **http://*IP_address_of_the_server*** 에 연결합니다. 웹 서버는 **/usr/share/nginx/html/index.html** 파일의 예제 콘텐츠를 표시합니다.

2.3. NGINX 웹 서버에 TLS 암호화 추가

이 섹션에서는 **example.com** 도메인의 NGINX 웹 서버에서 TLS 암호화를 활성화하는 방법을 설명합니다.

사전 요구 사항

- NGINX가 설치되어 있습니다.
- 개인 키는 **/etc/pki/tls/private/example.com.key** 파일에 저장됩니다.
  개인 키 및 CSR(인증서 서명 요청) 생성 및 CA(인증 기관)에서 인증서를 요청하는 방법에 대한 자세한 내용은 CA 문서를 참조하십시오.
- TLS 인증서는 **/etc/pki/tls/certs/example.com.crt** 파일에 저장됩니다. 다른 경로를 사용하는 경우 절차의 해당 단계를 조정합니다.
- CA 인증서가 서버의 TLS 인증서 파일에 추가되었습니다.

- 클라이언트 및 웹 서버는 서버의 호스트 이름을 웹 서버의 IP 주소로 확인합니다.
- 포트 **443** 은 로컬 방화벽에서 열려 있습니다.

절차

1. **/etc/nginx/nginx.conf** 파일을 편집하고 다음 **server** 블록을 구성의 **http** 블록에 추가합니다.

```
server {
    listen              443 ssl;
    server_name         example.com;
    root                /usr/share/nginx/html;
    ssl_certificate     /etc/pki/tls/certs/example.com.crt;
    ssl_certificate_key /etc/pki/tls/private/example.com.key;
}
```

2. 보안상의 이유로 **root** 사용자만 개인 키 파일에 액세스할 수 있도록 구성합니다.

```
# chown root:root /etc/pki/tls/private/example.com.key
# chmod 600 /etc/pki/tls/private/example.com.key
```

> **주의**
>
> 권한이 없는 사용자가 개인 키에 액세스한 경우 인증서를 취소하고 새 개인 키를 만들고 새 인증서를 요청합니다. 그렇지 않으면 TLS 연결이 더 이상 안전하지 않습니다.

3. **nginx** 서비스를 다시 시작합니다.

```
# systemctl restart nginx
```

검증 단계

- 브라우저를 사용하고 **https://example.com**에 연결

추가 리소스

- RHEL의 TLS에 대한 보안 고려 사항

2.4. NGINX를 HTTP 트래픽의 역방향 프록시로 구성

HTTP 트래픽의 역방향 프록시로 작동하도록 NGINX 웹 서버를 구성할 수 있습니다. 예를 들어 이 기능을 사용하여 요청을 원격 서버의 특정 하위 디렉터리에 전달할 수 있습니다. 클라이언트 관점에서 클라이언트는 액세스하는 호스트에서 콘텐츠를 로드합니다. 그러나 NGINX는 원격 서버에서 실제 콘텐츠를 로드하여 클라이언트로 전달합니다.

이 절차에서는 웹 서버의 **/example** 디렉터리로 트래픽을 URL **로 전달하는 방법을 설명합니다.**

사전 요구 사항

- NGINX 설치
- 선택 사항: TLS 암호화는 역방향 프록시에서 활성화됩니다.

절차

1. /etc/nginx/nginx.conf 파일을 편집하고 역방향 프록시를 제공해야 하는 server 블록에 다음 설정을 추가합니다.

```
location /example {
    proxy_pass https://example.com;
}
```

location 블록은 NGINX가 /example 디렉터리의 모든 요청을 로 전달함을 정의합니다.

2. SELinux가 NGINX가 트래픽을 전달할 수 있도록 httpd_can_network_connect SELinux 부울 매개변수를 1 로 설정합니다.

```
# setsebool -P httpd_can_network_connect 1
```

3. nginx 서비스를 다시 시작합니다.

```
# systemctl restart nginx
```

검증 단계

- 브라우저를 사용하여 http://*host_name*/example에 연결하면 https://example.com 내용이 표시됩니다.

2.5. NGINX를 HTTP 로드 밸런서로 구성

NGINX 역방향 프록시 기능을 사용하여 트래픽을 로드 밸런싱할 수 있습니다. 이 절차에서는 NGINX를 활성 연결 수가 가장 적은 다른 서버에 요청을 보내는 HTTP 로드 밸런서 장치로 구성하는 방법을 설명합니다. 두 서버를 모두 사용할 수 없는 경우 절차에서는 폴백 이유로 세 번째 호스트도 정의합니다.

사전 요구 사항

- NGINX 설치

절차

1. /etc/nginx/nginx.conf 파일을 편집하고 다음 설정을 추가합니다.

```
http {
    upstream backend {
        least_conn;
        server server1.example.com;
        server server2.example.com;
        server server3.example.com backup;
    }

    server {
        location / {
            proxy_pass http://backend;
        }
    }
}
```

backend 라는 호스트 그룹의 least_conn 지시문은 활성 연결 수가 가장 적은 호스트에 따라 NGINX가 server1.example.com 또는 server2.example.com에 요청을 보냅니다. NGINX에서는 다른 두 호스트를 사용할 수 없는 경우에만 server3.example.com을 백업으로 사용합니다.

proxy_pass 지시문을 http://backend 로 설정하면 NGINX는 역방향 프록시 역할을 하며 backend 호스트 그룹을 사용하여 이 그룹의 설정에 따라 요청을 분산합니다.

least_conn 로드 밸런싱 방법 대신 다음을 지정할 수 있습니다.

- 라운드 로빈을 사용하고 서버 간에 요청을 균등하게 배포하는 방법은 없습니다.
- IPv4 주소의 처음 세 옥텟 또는 클라이언트의 전체 IPv6 주소에서 계산된 해시를 기반으로 한 클라이언트 주소의 요청을 동일한 서버로 전송하는 ip_hash.
- 문자열, 변수 또는 둘 다 조합일 수 있는 사용자 정의 키를 기반으로 서버를 결정하는 해시 입니다. 일관된 매개 변수는 NGINX가 사용자 정의 해시 키 값을 기반으로 모든 서버에 요청을 배포하도록 구성합니다.

- 무작위 로 선택된 서버로 요청을 보내는 임의 실행.

2. **nginx** 서비스를 다시 시작합니다.

```
# systemctl restart nginx
```

2.6. 추가 리소스

- 공식 NGINX 문서. Red Hat은 이 문서를 유지 관리하지 않으며 설치한 NGINX 버전에서 작동하지 않을 수 있습니다.

- PKCS #11을 통해 암호화 하드웨어를 사용하도록 애플리케이션 구성.

3장. SQUID 캐싱 프록시 서버 구성

Squid는 콘텐츠를 캐시하여 대역폭과 부하 웹 페이지를 더 빠르게 줄이는 프록시 서버입니다. 이 장에서는 HTTP, HTTPS 및 FTP 프로토콜에 대한 프록시로 Squid를 설정하는 방법과 인증 및 액세스 제한에 대해 설명합니다.

3.1. 인증 없이 캐싱 프록시로 SQUID 설정

이 섹션에서는 인증 없이 캐싱 프록시로 Squid의 기본 구성에 대해 설명합니다. 이 절차에서는 IP 범위를 기반으로 프록시에 대한 액세스를 제한합니다.

사전 요구 사항

- 이 절차에서는 /etc/squid/squid.conf 파일이 squid 패키지에서 제공하는 것으로 가정합니다. 이전에 이 파일을 편집한 경우 파일을 제거하고 패키지를 다시 설치합니다.

절차

1. squid 패키지를 설치합니다.

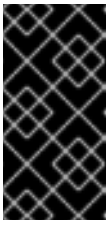
```
# yum install squid
```

2. /etc/squid/squid.conf 파일을 편집합니다.

   a. 프록시를 사용할 수 있어야 하는 IP 범위와 일치하도록 localnet ACL(액세스 제어 목록)을 조정합니다.

```
acl localnet src 192.0.2.0/24
acl localnet 2001:db8:1::/64
```

   기본적으로 /etc/squid/squid.conf 파일에는 localnet ACL에 지정된 모든 IP 범위의 프록시를 사용할 수 있는 http_access allow localnet 규칙이 포함되어 있습니다. http_access allow localnet 규칙 전에 모든 localnet ACL을 지정해야 합니다.



중요

사용자 환경과 일치하지 않는 기존 **acl localnet** 항목을 모두 제거합니다.

b.

다음 **ACL**은 기본 구성에 존재하며 **HTTPS** 프로토콜을 사용하는 포트로 **443** 을 정의합니다.

```
acl SSL_ports port 443
```

사용자가 다른 포트에서도 **HTTPS** 프로토콜을 사용할 수 있어야 하는 경우 다음 각 포트에 대한 **ACL**을 추가합니다.

```
acl SSL_ports port port_number
```

c.

acl Safe_ports 규칙 목록을 업데이트하여 **Squid**가 연결을 설정할 수 있는 포트를 구성합니다. 예를 들어 프록시를 사용하여 클라이언트가 포트 **21(FTP), 80(HTTP)** 및 **443(HTTPS)**의 리소스에만 액세스할 수 있도록 구성하려면 구성의 다음 **acl Safe_ports** 문만 유지합니다.

```
acl Safe_ports port 21
acl Safe_ports port 80
acl Safe_ports port 443
```

기본적으로 구성에는 **Safe_ports ACL**에 정의되지 않은 포트에 대한 액세스 거부를 정의하는 **http_access deny! Safe_ports** 규칙이 포함됩니다.

d.

cache_dir 매개변수의 캐시 유형, 캐시 크기 및 추가 캐시 유형별 설정을 구성합니다.

```
cache_dir ufs /var/spool/squid 10000 16 256
```

다음 설정이 필요합니다.

- **squid**는 **ufs** 캐시 유형을 사용합니다.
- **squid**는 해당 캐시를 **/var/spool/squid/** 디렉터리에 저장합니다.

- 캐시가 **10000 MB**까지 증가합니다.
- **squid**는 **/var/spool/squid/** 디렉터리에 **16 level-1** 하위 디렉터리를 생성합니다.
- **squid**는 각 **level-1** 디렉토리에 **256** 개의 하위 디렉터리를 생성합니다.

cache_dir 지시문을 설정하지 않으면 **Squid**가 캐시를 메모리에 저장합니다.

3. **cache_dir** 매개변수에서 **/var/spool/squid/** 와 다른 캐시 디렉터리를 설정하는 경우:

a. 캐시 디렉토리를 생성합니다.

```
# mkdir -p path_to_cache_directory
```

b. 캐시 디렉토리에 대한 권한을 구성합니다.

```
# chown squid:squid path_to_cache_directory
```

c. 강제 모드에서 **SELinux**를 실행하는 경우 캐시 디렉터리에 **squid_cache_t** 컨텍스트를 설정합니다.

```
# semanage fcontext -a -t squid_cache_t "path_to_cache_directory(/.*)?"
# restorecon -Rv path_to_cache_directory
```

시스템에서 **semanage** 유틸리티를 사용할 수 없는 경우 **policycoreutils-python-utils** 패키지를 설치합니다.

4. 방화벽에서 **3128** 포트를 엽니다.

```
# firewall-cmd --permanent --add-port=3128/tcp
# firewall-cmd --reload
```

5. **squid** 서비스를 활성화하고 시작합니다.

```
# systemctl enable --now squid
```

검증 단계

프록시가 올바르게 작동하는지 확인하려면 **curl** 유틸리티를 사용하여 웹 페이지를 다운로드합니다.

```
# curl -O -L "https://www.redhat.com/index.html" -x "proxy.example.com:3128"
```

curl 에 오류가 표시되지 않고 **index.html** 파일이 현재 디렉터리로 다운로드된 경우 프록시가 작동합니다.

3.2. LDAP 인증을 사용하여 SQUID를 캐싱 프록시로 설정

이 섹션에서는 **LDAP**를 사용하여 사용자를 인증하는 캐싱 프록시로 **Squid**의 기본 구성에 대해 설명합니다. 이 절차에서는 인증된 사용자만 프록시를 사용할 수 있도록 구성됩니다.

사전 요구 사항

- 이 절차에서는 **/etc/squid/squid.conf** 파일이 **squid** 패키지에서 제공하는 것으로 가정합니다. 이전에 이 파일을 편집한 경우 파일을 제거하고 패키지를 다시 설치합니다.
- **LDAP** 디렉터리에 **uid=proxy_user,cn=users,cn=accounts,dc=example,dc=com** 과 같은 서비스 사용자가 있습니다. **Squid**는 이 계정을 인증 사용자를 검색하기 위해서만 사용합니다. 인증 사용자가 있는 경우 **Squid**는 이 사용자로 을 디렉터리에 바인딩하여 인증을 확인합니다.

절차

1. **squid** 패키지를 설치합니다.

```
# yum install squid
```

2. **/etc/squid/squid.conf** 파일을 편집합니다.

   a. **basic_ldap_auth helper** 유틸리티를 구성하려면 **/etc/squid/squid.conf** 상단에 다음 설정 항목을 추가합니다.

```
auth_param basic program /usr/lib64/squid/basic_ldap_auth -b
"cn=users,cn=accounts,dc=example,dc=com" -D
```

> *"uid=proxy_user,cn=users,cn=accounts,dc=example,dc=com" -W /etc/squid/ldap_password -f "(&(objectClass=person)(uid=%s))" -ZZ -H ldap://ldap_server.example.com:389*

다음은 위의 예제에서 **basic_ldap_auth** 도우미 유틸리티에 전달된 매개변수를 설명합니다.

- **-B *base_DN*** 은 **LDAP** 검색 기반을 설정합니다.
- **-d *proxy_service_user_DN*** 은 **Squid**가 디렉터리에서 인증 사용자를 검색하는 데 사용하는 계정의 **DN**(고유 이름)을 설정합니다.
- **-W *path_to_password_file*** 은 프록시 서비스 사용자의 암호가 포함된 파일의 경로를 설정합니다. 암호 파일을 사용하면 운영 체제의 프로세스 목록에 암호가 표시되지 않습니다.
- **-F *LDAP_filter*** 는 **LDAP** 검색 필터를 지정합니다. **squid**는 인증 사용자가 제공한 사용자 이름으로 **%s** 변수를 대체합니다.

  예제의 **(&(objectClass=person)(uid=%s)** 필터는 사용자 이름이 **uid** 특성에 설정된 값과 디렉터리 항목에 사람 오브젝트 클래스가 포함되어 있어야 함을 정의합니다.
- **-ZZ** 는 **STARTTLS** 명령을 사용하여 **LDAP** 프로토콜을 통해 **TLS** 암호화 연결을 적용합니다. 다음과 같은 경우에는 **-ZZ** 를 생략하십시오.
  - **LDAP** 서버는 암호화된 연결을 지원하지 않습니다.
  - **URL**에 지정된 포트는 **LDAPS** 프로토콜을 사용합니다.
- **H LDAP_URL** 매개 변수는 프로토콜, 호스트 이름 또는 **IP** 주소, **LDAP** 서버의 포트를 **URL** 형식으로 지정합니다.

b. **Squid**가 인증된 사용자만 프록시를 사용하도록 허용하는 다음 **ACL** 및 규칙을 추가합니다.

```
acl ldap-auth proxy_auth REQUIRED
http_access allow ldap-auth
```

중요

http_access deny 모든 규칙을 거부하기 전에 이러한 설정을 지정합니다.

c.

localnet ACL에 지정된 **IP** 범위에서 프록시 인증을 바이패스하려면 다음 규칙을 제거합니다.

```
http_access allow localnet
```

d.

다음 **ACL**은 기본 구성에 존재하며 **HTTPS** 프로토콜을 사용하는 포트로 **443** 을 정의합니다.

```
acl SSL_ports port 443
```

사용자가 다른 포트에서도 **HTTPS** 프로토콜을 사용할 수 있어야 하는 경우 다음 각 포트에 대한 **ACL**을 추가합니다.

```
acl SSL_ports port port_number
```

e.

acl Safe_ports 규칙 목록을 업데이트하여 **Squid**가 연결을 설정할 수 있는 포트를 구성합니다. 예를 들어 프록시를 사용하여 클라이언트가 포트 **21(FTP), 80(HTTP)** 및 **443(HTTPS)**의 리소스에만 액세스할 수 있도록 구성하려면 구성의 다음 **acl Safe_ports** 문만 유지합니다.

```
acl Safe_ports port 21
acl Safe_ports port 80
acl Safe_ports port 443
```

기본적으로 구성에는 **Safe_ports ACL**에 정의되지 않은 포트에 대한 액세스 거부를 정의하는 **http_access deny! Safe_ports** 규칙이 포함됩니다.

f.

cache_dir 매개변수의 캐시 유형, 캐시 크기 및 추가 캐시 유형별 설정을 구성합니다.

```
cache_dir ufs /var/spool/squid 10000 16 256
```

다음 설정이 필요합니다.

- **squid**는 **ufs** 캐시 유형을 사용합니다.
- **squid**는 해당 캐시를 **/var/spool/squid/** 디렉터리에 저장합니다.
- 캐시가 **10000 MB**까지 증가합니다.
- **squid**는 **/var/spool/squid/** 디렉터리에 **16 level-1** 하위 디렉터리를 생성합니다.
- **squid**는 각 **level-1** 디렉토리에 **256** 개의 하위 디렉터리를 생성합니다.

  cache_dir 지시문을 설정하지 않으면 **Squid**가 캐시를 메모리에 저장합니다.

3. **cache_dir** 매개변수에서 **/var/spool/squid/** 와 다른 캐시 디렉터리를 설정하는 경우:

   a. 캐시 디렉토리를 생성합니다.

   ```
   # mkdir -p path_to_cache_directory
   ```

   b. 캐시 디렉토리에 대한 권한을 구성합니다.

   ```
   # chown squid:squid path_to_cache_directory
   ```

   c. 강제 모드에서 **SELinux**를 실행하는 경우 캐시 디렉터리에 **squid_cache_t** 컨텍스트를 설정합니다.

   ```
   # semanage fcontext -a -t squid_cache_t "path_to_cache_directory(/.*)?"
   # restorecon -Rv path_to_cache_directory
   ```

   시스템에서 **semanage** 유틸리티를 사용할 수 없는 경우 **policycoreutils-python-utils** 패키지를 설치합니다.

4. **LDAP** 서비스 사용자의 암호를 **/etc/squid/ldap_password** 파일에 저장하고 파일에 대한 적절한 권한을 설정합니다.

```
# echo "password" > /etc/squid/ldap_password
# chown root:squid /etc/squid/ldap_password
# chmod 640 /etc/squid/ldap_password
```

5. 방화벽에서 **3128** 포트를 엽니다.

```
# firewall-cmd --permanent --add-port=3128/tcp
# firewall-cmd --reload
```

6. **squid** 서비스를 활성화하고 시작합니다.

```
# systemctl enable --now squid
```

검증 단계

프록시가 올바르게 작동하는지 확인하려면 **curl** 유틸리티를 사용하여 웹 페이지를 다운로드합니다.

```
# curl -O -L "https://www.redhat.com/index.html" -x
"user_name:password@proxy.example.com:3128"
```

curl에서 오류를 표시하지 않고 **index.html** 파일이 현재 디렉터리로 다운로드된 경우 프록시가 작동합니다.

문제 해결 단계

도우미 유틸리티가 올바르게 작동하는지 확인하려면 다음을 수행하십시오.

1. **auth_param** 매개변수에 사용한 것과 동일한 설정으로 도우미 유틸리티를 수동으로 시작합니다.

```
# /usr/lib64/squid/basic_ldap_auth -b "cn=users,cn=accounts,dc=example,dc=com" -D
"uid=proxy_user,cn=users,cn=accounts,dc=example,dc=com" -W
/etc/squid/ldap_password -f "(&(objectClass=person)(uid=%s))" -ZZ -H
ldap://ldap_server.example.com:389
```

2. 유효한 사용자 이름과 암호를 입력하고 **Enter** 키를 누릅니다.

```
user_name password
```

도우미 유틸리티에서 **OK** 를 반환하는 경우 인증이 성공했습니다.

3.3. KERBEROS 인증을 사용하여 캐싱 프록시로 SQUID 설정

이 섹션에서는 **Kerberos**를 사용하여 사용자를 **AD(Active Directory)** 인증하는 캐싱 프록시로 **Squid** 의 기본 구성에 대해 설명합니다. 이 절차에서는 인증된 사용자만 프록시를 사용할 수 있도록 구성됩니다.

사전 요구 사항

- 이 절차에서는 **/etc/squid/squid.conf** 파일이 **squid** 패키지에서 제공하는 것으로 가정합니다. 이전에 이 파일을 편집한 경우 파일을 제거하고 패키지를 다시 설치합니다.
- **Squid**를 설치할 서버는 **AD** 도메인의 멤버입니다. 자세한 내용은 **Red Hat Enterprise Linux 8의 도메인 멤버로 Samba 설정** 문서 를 참조하십시오.

절차

1. 다음 패키지를 설치합니다.

```
# yum install squid krb5-workstation
```

2. **AD** 도메인 관리자로 인증합니다.

```
# kinit administrator@AD.EXAMPLE.COM
```

3. **Squid**의 키탭을 생성하여 **/etc/squid/HTTP.keytab** 파일에 저장하십시오.

```
# export KRB5_KTNAME=FILE:/etc/squid/HTTP.keytab
# net ads keytab CREATE -U administrator
```

4. **keytab**에 **HTTP** 서비스 주체를 추가합니다.

```
# net ads keytab ADD HTTP -U administrator
```

5. **keytab** 파일의 소유자를 **squid** 사용자로 설정합니다.

```
# chown squid /etc/squid/HTTP.keytab
```

6. 필요한 경우 **keytab** 파일에 프록시 서버의 **FQDN**(정규화된 도메인 이름)에 대한 **HTTP** 서비스 주체가 포함되어 있는지 확인합니다.

```
 klist -k /etc/squid/HTTP.keytab
Keytab name: FILE:/etc/squid/HTTP.keytab
KVNO Principal
---- --------------------------------------------------
...
   2 HTTP/proxy.ad.example.com@AD.EXAMPLE.COM
...
```

7. **/etc/squid/squid.conf** 파일을 편집합니다.

   a. **negotiate_kerberos_auth** 도우미 유틸리티를 구성하려면 **/etc/squid/squid.conf** 의 상단에 다음 설정 항목을 추가하십시오.

```
auth_param negotiate program /usr/lib64/squid/negotiate_kerberos_auth -k
/etc/squid/HTTP.keytab -s HTTP/proxy.ad.example.com@AD.EXAMPLE.COM
```

   다음은 위의 예제에서 **negotiate_kerberos_auth** 도우미 유틸리티로 전달되는 매개변수를 설명합니다.

   - **-K** *파일은* 키 탭 파일의 경로를 설정합니다. **squid** 사용자는 이 파일에 대한 읽기 권한이 있어야 합니다.
   - **-s HTTP/***host_name@kerberos_realm* 은 **Squid**가 사용하는 **Kerberos** 주체를 설정합니다.

   선택적으로 다음 매개변수 중 하나 또는 둘 다를 도우미 유틸리티에 전달하여 로깅을 활성화할 수 있습니다.

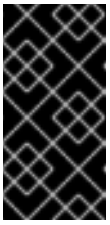
   - **-** 인증 사용자와 같은 정보 메시지를 기록합니다.

- **-D**는 디버그 로깅을 활성화합니다.

squid는 도우미 유틸리티의 디버깅 정보를 **/var/log/squid/cache.log** 파일에 기록합니다.

b. **Squid**가 인증된 사용자만 프록시를 사용하도록 허용하는 다음 **ACL** 및 규칙을 추가합니다.

```
acl kerb-auth proxy_auth REQUIRED
http_access allow kerb-auth
```



중요

http_access deny 모든 규칙을 거부하기 전에 이러한 설정을 지정합니다.

c. **localnet ACL**에 지정된 **IP** 범위에서 프록시 인증을 바이패스하려면 다음 규칙을 제거합니다.

```
http_access allow localnet
```

d. 다음 **ACL**은 기본 구성에 존재하며 **HTTPS** 프로토콜을 사용하는 포트로 **443** 을 정의합니다.

```
acl SSL_ports port 443
```

사용자가 다른 포트에서도 **HTTPS** 프로토콜을 사용할 수 있어야 하는 경우 다음 각 포트에 대한 **ACL**을 추가합니다.

```
acl SSL_ports port port_number
```

e. **acl Safe_ports** 규칙 목록을 업데이트하여 **Squid**가 연결을 설정할 수 있는 포트를 구성합니다. 예를 들어 프록시를 사용하여 클라이언트가 포트 **21(FTP), 80(HTTP)** 및 **443(HTTPS)**의 리소스에만 액세스할 수 있도록 구성하려면 구성의 다음 **acl Safe_ports** 문만 유지합니다.

```
acl Safe_ports port 21
acl Safe_ports port 80
acl Safe_ports port 443
```

-

기본적으로 구성에는 **Safe_ports ACL**에 정의되지 않은 포트에 대한 액세스 거부를 정의하는 **http_access deny! Safe_ports** 규칙이 포함됩니다.

f.

cache_dir 매개변수의 캐시 유형, 캐시 크기 및 추가 캐시 유형별 설정을 구성합니다.

```
cache_dir ufs /var/spool/squid 10000 16 256
```

다음 설정이 필요합니다.

- **squid**는 **ufs** 캐시 유형을 사용합니다.
- **squid**는 해당 캐시를 **/var/spool/squid/** 디렉터리에 저장합니다.
- 캐시가 **10000 MB**까지 증가합니다.
- **squid**는 **/var/spool/squid/** 디렉터리에 **16 level-1** 하위 디렉터리를 생성합니다.
- **squid**는 각 **level-1** 디렉토리에 **256** 개의 하위 디렉터리를 생성합니다.

cache_dir 지시문을 설정하지 않으면 **Squid**가 캐시를 메모리에 저장합니다.

8.

cache_dir 매개변수에서 **/var/spool/squid/** 와 다른 캐시 디렉터리를 설정하는 경우:

a.

캐시 디렉토리를 생성합니다.

```
# mkdir -p path_to_cache_directory
```

b.

캐시 디렉토리에 대한 권한을 구성합니다.

```
# chown squid:squid path_to_cache_directory
```

c.

강제 모드에서 **SELinux**를 실행하는 경우 캐시 디렉터리에 **squid_cache_t** 컨텍스트를 설정합니다.

```
# semanage fcontext -a -t squid_cache_t "path_to_cache_directory(/.*)?"
# restorecon -Rv path_to_cache_directory
```

시스템에서 **semanage** 유틸리티를 사용할 수 없는 경우 **policycoreutils-python-utils** 패키지를 설치합니다.

9. 방화벽에서 **3128** 포트를 엽니다.

```
# firewall-cmd --permanent --add-port=3128/tcp
# firewall-cmd --reload
```

10. **squid** 서비스를 활성화하고 시작합니다.

```
# systemctl enable --now squid
```

검증 단계

프록시가 올바르게 작동하는지 확인하려면 **curl** 유틸리티를 사용하여 웹 페이지를 다운로드합니다.

```
# curl -O -L "https://www.redhat.com/index.html" --proxy-negotiate -u : -x
"proxy.ad.example.com:3128"
```

curl 에서 오류를 표시하지 않고 **index.html** 파일이 현재 디렉터리에 있는 경우 프록시가 작동합니다.

문제 해결 단계

Kerberos 인증을 수동으로 테스트하려면 다음을 수행합니다.

1. **AD** 계정에 대한 **Kerberos** 티켓을 받습니다.

```
# kinit user@AD.EXAMPLE.COM
```

2. 선택적으로 티켓을 표시합니다.

```
# klist
```

3. **negotiate_kerberos_auth_test** 유틸리티를 사용하여 인증을 테스트합니다.

```
# /usr/lib64/squid/negotiate_kerberos_auth_test proxy.ad.example.com
```

도우미 유틸리티에서 토큰을 반환하면 인증이 성공했습니다.

```
Token: YIIFtAYGKwYBBQUCoIIFqDC...
```

3.4. SQUID에서 도메인 거부 목록 구성

관리자는 특정 도메인에 대한 액세스를 차단하려는 경우가 많습니다. 이 섹션에서는 Squid에서 도메인 거부 목록을 구성하는 방법을 설명합니다.

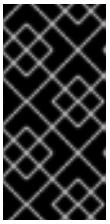
사전 요구 사항

- Squid가 구성되어 있으며 사용자는 프록시를 사용할 수 있습니다.

절차

1. **/etc/squid/squid.conf** 파일을 편집하고 다음 설정을 추가합니다.

```
acl domain_deny_list dstdomain "/etc/squid/domain_deny_list.txt"
http_access deny all domain_deny_list
```

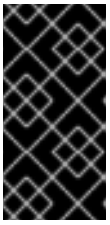


중요

사용자 또는 클라이언트에 대한 액세스를 허용하는 첫 번째 **http_access allow** 문 앞에 이러한 항목을 추가합니다.

2. **/etc/squid/domain_deny_list.txt** 파일을 만들고 차단하려는 도메인을 추가합니다. 예를 들어 하위 도메인을 포함한 **example.com** 에 대한 액세스를 차단하고 **example.net** 을 차단하려면 다음을 추가합니다.

```
.example.com
example.net
```



중요

squid 구성에서 **/etc/squid/domain_deny_list.txt** 파일을 참조하는 경우 이 파일은 비워서는 안 됩니다. 파일이 비어 있으면 **Squid**가 시작되지 않습니다.

3. **squid** 서비스를 다시 시작하십시오.

```
# systemctl restart squid
```

3.5. 특정 포트 또는 IP 주소에서 수신 대기하도록 SQUID 서비스 구성

기본적으로 **Squid** 프록시 서비스는 모든 네트워크 인터페이스의 **3128** 포트에서 수신 대기합니다. 이 섹션에서는 포트를 변경하고 특정 **IP** 주소에서 수신 대기하도록 **Squid**를 구성하는 방법에 대해 설명합니다.

사전 요구 사항

- **squid** 패키지가 설치되어 있습니다.

절차

1. **/etc/squid/squid.conf** 파일을 편집합니다.

   - **Squid** 서비스가 수신 대기하는 포트를 설정하려면 **http_port** 매개 변수에 포트 번호를 설정합니다. 예를 들어 포트를 **8080** 으로 설정하려면 다음을 설정합니다.

     ```
     http_port 8080
     ```

   - **Squid** 서비스가 수신 대기하는 **IP** 주소를 구성하려면 **http_port** 매개변수에서 **IP** 주소 및 포트 번호를 설정합니다. 예를 들어 **Squid**가 포트 **3128** 의 **192.0.2.1 IP** 주소에서만 수신 대기하도록 구성하려면 다음을 설정합니다.

     ```
     http_port 192.0.2.1:3128
     ```

     구성 파일에 여러 **http_port** 매개 변수를 추가하여 **Squid**가 여러 포트 및 **IP** 주소에서 수신 대기하도록 구성합니다.

```
http_port 192.0.2.1:3128
http_port 192.0.2.1:8080
```

2. Squid가 기본값(3128)으로 다른 포트를 사용하도록 구성한경우:

   a. 방화벽에서 포트를 엽니다.

   ```
   # firewall-cmd --permanent --add-port=port_number/tcp
   # firewall-cmd --reload
   ```

   b. 강제 모드에서 SELinux를 실행하는 경우 포트를 squid_port_t 포트 유형 정의에 할당합니다.

   ```
   # semanage port -a -t squid_port_t -p tcp port_number
   ```

   시스템에서 semanage 유틸리티를 사용할 수 없는 경우 policycoreutils-python-utils 패키지를 설치합니다.

3. squid 서비스를 다시 시작하십시오.

```
# systemctl restart squid
```

3.6. 추가 리소스

- 설정 매개변수 ECDHE/share/doc/squid-<version>/squid.conf.documented