# Red Hat Enterprise Linux 9

## 部署 Web 服务器和反向代理

在 Red Hat Enterprise Linux 9 中设置和配置 Web 服务器和反向代理

Last Updated: 2024-06-26

# Red Hat Enterprise Linux 9 部署 Web 服务器和反向代理

在 Red Hat Enterprise Linux 9 中设置和配置 Web 服务器和反向代理

## 法律通告



## 摘要

在 Red Hat Enterprise Linux 9 上配置并运行 Apache HTTP web 服务器、NGINX web 服务器或 Squid 缓存代理服务器。配置 TLS 加密。为 Apache HTTP web 服务器配置 Kerberos 身份验证。将 NGINX 设置为 HTTP 流量的反向代理或 HTTP 负载均衡器。将 Squid 配置为没有身份验证、具有 LDAP 身份验证，或具有 Kerberos 身份验证的缓存代理。

# 目录

# 对红帽文档提供反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。

**通过 Jira 提交反馈（需要帐户）**

1. 登录到 Jira 网站。
2. 在顶部导航栏中点 **Create**
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您对改进的建议。包括文档相关部分的链接。
5. 点对话框底部的 **Create**。

# 第 1 章 设置 APACHE HTTP WEB 服务器

## 1.1. APACHE HTTP WEB 服务器简介

*Web 服务器*是一个通过 Web 向客户端提供内容的网络服务。这通常是网页，但也可以提供任何其他文档。Web 服务器也称为 HTTP 服务器，因为它们使用 *超文本传输协议* (**HTTP**)。

**Apache HTTP 服务器 httpd** 是由 Apache Software Foundation 开发的开源 Web 服务器。

如果您要从之前的 Red Hat Enterprise Linux 版本升级，您必须相应地更新 **httpd** 服务配置。本节介绍了一些新添加的功能，并指导您完成之前的配置文件的更新。

## 1.2. APACHE HTTP 服务器中的显著变化

RHEL 9 提供 Apache HTTP 服务器的版本 2.4.48。RHEL 8 发布的 2.4.37 版本的显著变化包括：

- Apache HTTP 服务器控制接口(**apachectl**)：
  - 现在，**apachectl status** 输出禁用了 **systemctl** pager。
  - 现在，如果您传递了附加参数，则 **apachectl** 命令会失败，而不是发出警告。
  - **apachectl graceful-stop** 命令现在会立即返回。
  - **apachectl configtest** 命令现在在不更改 SELinux 上下文的情况下执行 **httpd -t** 命令。
  - RHEL 中的 **apachectl(8)** man page 现在完全指明了与上游 **apachectl** 之间的差异。
- Apache eXtenSion 工具(**pxs**)：
  - 构建 **httpd** 软件包时，**/usr/bin/apxs** 命令不再使用或公开编译器选择的标志。现在，您可以使用 **/usr/lib64/httpd/build/vendor-apxs** 命令应用与构建 **httpd** 相同的编译器标志。要使用 **vendor-apxs** 命令，您必须首先安装 **redhat-rpm-config** 软件包。
- Apache 模块：
  - **mod_lua** 模块现在在一个单独的软件包中提供。
  - PHP 提供的与 Apache HTTP 服务器一起使用的 **mod_php** 模块已被删除。从 RHEL 8 开始，PHP 脚本默认使用 FastCGI Process Manager (**php-fpm**)运行。如需更多信息，请参阅 将 PHP 与 Apache HTTP 服务器一起使用 。
- 配置语法更改：
  - 在由 **mod_access_compat** 模块提供的已弃用的 **Allow** 指令中，注释（ **#** 字符）现在会触发语法错误，而不是静默忽略。
- 其他更改：
  - 内核线程 ID 现在直接在错误信息中使用，从而使它们准确且更简洁。
  - 多个小幅改进和漏洞修复。
  - 几个新接口对模块作者可用。

从 RHEL 8 开始，**httpd** 模块 API 没有向后兼容的更改。

Apache HTTP Server 2.4 是此 Application Stream 的初始版本，您可以将其作为 RPM 软件包轻松安装。

## 1.3. APACHE 配置文件

默认情况下，**httpd** 在启动后读取配置文件。您可以在下表中查看配置文件的位置列表。

**表 1.1. httpd 服务配置文件**

| 路径 | 描述 |
| --- | --- |
| **/etc/httpd/conf/httpd.conf** | 主配置文件。 |
| **/etc/httpd/conf.d/** | 主配置文件中包含的配置文件的辅助目录。 |
| **/etc/httpd/conf.modules.d/** | 用于载入 Red Hat Enterprise Linux 中打包动态模块的配置文件的辅助目录。在默认配置中，首先会处理这些配置文件。 |

虽然默认配置适用于大多数情况，但您也可以使用其他配置选项。要让任何更改生效，请首先重启 web 服务器。

要检查配置中的可能错误，在 shell 提示符后输入以下内容：

```
# apachectl configtest
Syntax OK
```

要更方便地从错误中恢复，请在编辑前复制原始文件。

## 1.4. 管理 HTTPD 服务

本节描述了如何启动、停止和重新启动 **httpd** 服务。

**先决条件**

- 已安装 Apache HTTP 服务器。

**流程**

- 要启动 **httpd** 服务，请输入：

```
# systemctl start httpd
```

- 要停止 **httpd** 服务，请输入：

```
# systemctl stop httpd
```

- 要重启 **httpd** 服务，请输入：

```
# systemctl restart httpd
```

## 1.5. 设置单实例 APACHE HTTP 服务器

您可以设置一个单实例 Apache HTTP 服务器来提供静态 HTML 内容。

如果 Web 服务器应该为与服务器关联的所有域提供相同的内容，请按照以下流程操作。如果要为不同的域提供不同的内容，请设置基于名称的虚拟主机。详情请参阅 配置 Apache 基于名称的虚拟主机。

**流程**

1. 安装 **httpd** 软件包：

```
# dnf install httpd
```

2. 如果使用 **firewalld**，请在本地防火墙中打开 TCP 端口 **80** ：

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

3. 启用并启动 **httpd** 服务：

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

4. 可选：将 HTML 文件添加到 **/var/www/html/** 目录中。

**注意**

在 向**/var/www/html/** 添加内容时，在**httpd**默认运行的情况下，文件和目录必须可被用户读取。内容所有者可以是 **root**用户和**root**用户组，也可以是管理员所选择的其他用户或组。如果内容所有者是 **root** 用户和 **root** 用户组，则文件必须可被其他用户读取。所有文件和目录的 SELinux 上下文必须为 **httpd_sys_content_t**，其默认应用于 **/var/www** 目录中的所有内容。

**验证步骤**

- 使用 Web 浏览器连接到 **http://*server_IP_or_host_name*/**。
  如果 **/var/www/html/** 目录为空，或者不包含 **index.html**或**index.htm**文件，则 Apache 会显示 **Red Hat Enterprise Linux 测试页面**。如果 **/var/www/html/** 包含具有不同名称的 HTML 文件，您可以通过输入该文件的 URL 来加载它们，如 **http://*server_IP_or_host_name*/*example.html***。

**其它资源**

- Apache 手册：安装 Apache HTTP 服务器手册。
- 请参见 **httpd.service(8)** 手册页。

## 1.6. 配置基于 APACHE 名称的虚拟主机

基于名称的虚拟主机可让 Apache 为解析到服务器 IP 地址的不同域提供不同的内容。

您可以使用单独的文档根目录为 **example.com** 和 **example.net** 域设置虚拟主机。两个虚拟主机都提供静态 HTML 内容。

**先决条件**

**先决条件**

- 客户端和 Web 服务器将 **example.com** 和 **example.net** 域解析为 Web 服务器的 IP 地址。
  请注意，您必须手动将这些条目添加到 DNS 服务器中。

**流程**

1. 安装 **httpd** 软件包：

```
# dnf install httpd
```

2. 编辑 **/etc/httpd/conf/httpd.conf** 文件：

   a. 为 **example.com** 域添加以下虚拟主机配置：

```
<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>
```

   这些设置配置以下内容：

   - **<VirtualHost *:80>** 指令中的所有设置都是针对这个虚拟主机的。
   - **DocumentRoot** 设置虚拟主机的 Web 内容的路径。
   - **ServerName** 设置此虚拟主机为其提供内容服务的域。
     要设置多个域，请在配置中添加 **ServerAlias** 参数，并在此参数中指定用空格分开的额外域。
   - **CustomLog** 设置虚拟主机的访问日志的路径。
   - **ErrorLog** 设置虚拟主机错误日志的路径。

**注意**

Apache 还将配置中找到的第一个虚拟主机用于与**ServerName**和**Server Alias**参数中设置的任何域不匹配的请求。这还包括发送到服务器 IP 地址的请求。

3. 为 **example.net** 域添加类似的虚拟主机配置：

```
<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. 为两个虚拟主机创建文档根目录：

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

5. 如果您在 **DocumentRoot** 参数中设置的路径不在/**var**/**www**/中，请在两个文档根中设置 **httpd_sys_content_t** 上下文：

```
# 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/
```

这些命令在/**srv**/**example.com**/和/**srv**/ **example.net**/ 目录中设置 **httpd_sys_content_t**上下文。

请注意，您必须安装 **policycoreutils-python-utils** 软件包才能运行**restorecon** 命令。

6. 如果您使用 **firewalld**，请在本地防火墙中打开端口 **80** ：

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

7. 启用并启动 **httpd** 服务：

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

**验证步骤**

1. 在每个虚拟主机的文档 root 中创建不同的示例文件：

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

2. 使用浏览器并连接到 **http://example.com**Web 服务器显示**example.com**虚拟主机中的示例文件。

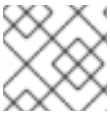
3. 使用浏览器并连接到 **http://example.net**Web 服务器显示**example.net**虚拟主机中的示例文件。

**其它资源**

- 安装 Apache HTTP 服务器手册 - 虚拟主机

## 1.7. 为 APACHE HTTP WEB 服务器配置 KERBEROS 验证

要在 Apache HTTP web 服务器中执行 Kerberos 身份验证，RHEL 9 使用 **mod_auth_gssapi** Apache 模块。Generic Security Services API(**GSSAPI**)是请求使用安全库（如 Kerberos）的应用程序的接口。**gssproxy** 服务允许对 **httpd** 服务器实施特权分离，从安全的角度来看，这优化了此过程。



**注意**

**mod_auth_gssapi** 模块取代了已删除的 **mod_auth_kerb** 模块。

**先决条件**

- 已安装了 **httpd**, **mod_auth_gssapi** 和 **gssproxy** 软件包。
- Apache Web 服务器已设置，并且 **httpd** 服务在运行。

### 1.7.1. 在 IdM 环境中设置 GSS-Proxy

这个流程描述了如何设置 **GSS-Proxy**，以便在 Apache HTTP Web 服务器中执行 Kerberos 身份验证。

**流程**

1. 通过创建服务主体来启用对 HTTP/<SERVER_NAME>@realm 主体的**keytab**文件的访问：

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

2. 检索存储在**/etc/gssproxy/http.keytab**文件中的主体的**keytab**：

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

此步骤将权限设置为 400，因此只有 **root** 用户有权访问 **keytab** 文件。**apache** 用户无法访问。

3. 使用以下内容创建 **/etc/gssproxy/80-httpd.conf** 文件：

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

4. 重启并启用 **gssproxy** 服务：

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

**其它资源**

- **gssproxy (8)** 手册页
- **gssproxy-mech(8)**手册页
- **gssproxy.conf (5)** 手册页

### 1.7.2. 为 Apache HTTP Web 服务器共享的目录配置 Kerberos 身份验证

这个过程描述了如何为 **/var/www/html/private/** 目录配置 Kerberos 身份验证。

**先决条件**

- **gssproxy** 服务已配置并在运行。

**流程**

1. 配置 **mod_auth_gssapi**模块来保护 **/var/www/html/private/**目录：

```
<Location /var/www/html/private>
  AuthType GSSAPI
  AuthName "GSSAPI Login"
```

```
  Require valid-user
</Location>
```

2. 创建系统单元配置置入文件：

```
# systemctl edit httpd.service
```

3. 在系统置入文件中添加以下参数：

```
[Service]
Environment=GSS_USE_PROXY=1
```

4. 重新载入**systemd**配置：

```
# systemctl daemon-reload
```

5. 重启**httpd**服务：

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

**验证步骤**

1. 获取Kerberos ticket：

```
# kinit
```

2. 在浏览器中打开到受保护目录的URL。

## 1.8. 在APACHE HTTP服务器上配置TLS加密

默认情况下，Apache 使用未加密的 HTTP 连接向客户端提供内容。这部分论述了如何在 Apache HTTP 服务器上启用 TLS 加密和配置常用的与加密相关的设置。

**先决条件**

- Apache HTTP 服务器已安装并运行。

### 1.8.1. 在 Apache HTTP 服务器中添加 TLS 加密

您可以在 Apache HTTP 服务器上为 **example.com** 域启用 TLS 加密。

**先决条件**

- Apache HTTP 服务器已安装并运行。
- 私钥存储在 **/etc/pki/tls/private/example.com.key** 文件中。
  有关创建私钥和证书签名请求(CSR)的详细信息，以及如何从证书颁发机构(CA)请求证书，请参阅您的 CA 文档。或者，如果您的 CA 支持 ACME 协议，您可以使用 **mod_md** 模块自动检索和调配 TLS 证书。
- TLS 证书存储在**/etc/pki/tls/certs/example.com.crt**文件中。如果您使用其他路径，请调整该流程的对应步骤。

- CA 证书存储在 **/etc/pki/tls/certs/ca.crt** 文件中。如果您使用其他路径，请调整该流程的对应步骤。
- 客户端和网页服务器会将服务器的主机名解析为 web 服务器的 IP 地址。
- 如果服务器运行 RHEL 9.2 或更高版本，并且启用了 FIPS 模式，则客户端必须支持 Extended Master Secret(EMS)扩展或使用 TLS 1.3。没有 EMS 的 TLS 1.2 连接会失败。如需更多信息，请参阅 强制 TLS 扩展"Extended Master Secret" 知识库文章。

**流程**

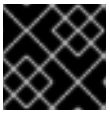
1. 安装 **mod_ssl** 软件包：

```
# dnf install mod_ssl
```

2. 编辑**/etc/httpd/conf.d/ssl.conf**文件，并将以下设置添加到 **<VirtualHost _default_:443>**指令中：

   a. 设置服务器名称：

```
ServerName example.com
```



**重要**

服务器名称必须与证书的 **Common Name**字段中设置的条目匹配。

   a. 可选：如果证书在 **Subject Alt Names** (SAN)字段中包含额外的主机名，您可以 配置**mod_ssl**来为这些主机名提供 TLS 加密。要配置此功能，请添加具有对应名称的**ServerAliases**参数：

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

   b. 设置到私钥、服务器证书和 CA 证书的路径：

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

3. 出于安全考虑，配置成只有 **root** 用户才可以访问私钥文件：

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

**警告**

如果私钥被设置为可以被未授权的用户访问，则需要撤销证书，然后再创建一个新私钥并请求一个新证书。否则，TLS 连接就不再安全。

4. 如果您使用 **firewalld**，请在本地防火墙中打开端口 **443** ：

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

5. 重启**httpd**服务：

```
# systemctl restart httpd
```

**注意**

如果您使用密码来保护私钥文件，则必须在每次 **httpd**服务启动时都输入此密码。

**验证步骤**

- 使用浏览器并连接到**https://*example.com***。

**其它资源**

- SSL/TLS 加密
- RHEL 9 中 TLS 的安全注意事项

### 1.8.2. 在 Apache HTTP 服务器中设置支持的 TLS 协议版本

默认情况下，RHEL 上的 Apache HTTP 服务器使用系统范围的加密策略来定义安全默认值，这些值也与最新的浏览器兼容。例如，**DEFAULT**策略定义了在 apache 中只启用 **TLSv1.2**和**TLSv1.3**协议版本。

您可以手动配置 Apache HTTP 服务器支持哪个 TLS 协议版本。如果您的环境只需要启用特定的 TLS 协议版本，请按照以下步骤操作，例如：

- 如果您的环境要求客户端也可以使用弱 **TLS1** (TLSv1.0)或**TLS1.1**协议。
- 如果你想将 Apache 配置为只支持**TLSv1.2**或**TLSv1.3**协议。

**先决条件**

- TLS 加密在服务器上启用，如 将 TLS 加密添加到 Apache HTTP 服务器 中所述。
- 如果服务器运行 RHEL 9.2 或更高版本，并且启用了 FIPS 模式，则客户端必须支持 Extended Master Secret(EMS)扩展或使用 TLS 1.3。没有 EMS 的 TLS 1.2 连接会失败。如需更多信息，请参阅 强制 TLS 扩展"Extended Master Secret" 知识库文章。

**流程**

1. 编辑 **/etc/httpd/conf/httpd.conf** 文件，并将以下设置添加到您要为其设置 TLS 协议版本的**<VirtualHost>**指令中。例如，只启用**TLSv1.3**协议：

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

2. 重启**httpd**服务：

```
# systemctl restart httpd
```

**验证步骤**

1. 使用以下命令来验证服务器是否支持**TLSv1.3**:

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

2. 使用以下命令来验证服务器是否不支持**TLSv1.2** ：

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

如果服务器不支持该协议，命令会返回一个错误：

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

3. 可选：重复用于其他 TLS 协议版本的命令。

**其它资源**

- **update-crypto-policies(8)** 手册页
- 使用系统范围的加密策略。
- 有关 **SSLProtocol** 参数的详情，请参考 Apache 手册中的 **mod_ssl** 文档：安装 Apache HTTP 服务器手册。

### 1.8.3. 在 Apache HTTP 服务器中设置支持的密码

默认情况下，Apache HTTP 服务器使用定义安全默认值的系统范围的加密策略，这些值也与最新的浏览器兼容。有关系统范围加密允许的密码列表，请查看**/etc/crypto-policies/back-ends/openssl.config** 文件。

您可以手动配置 Apache HTTP 服务器支持哪种密码。如果您的环境需要特定的加密系统，请按照以下步骤操作。

**先决条件**

- TLS 加密在服务器上启用，如 将 TLS 加密添加到 Apache HTTP 服务器 中所述。

**流程**

1. 编辑**/etc/httpd/conf/httpd.conf**文件，并将**SSLCipherSuite**参数添加到您要为其设置 TLS 密码的**<VirtualHost>**指令中：

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

这个示例只启用 **EECDH+AESGCM**、**EDH+AESGCM**、**AES256+EECDH** 和 **AES256+EDH**密码，并禁用所有使用**SHA1**和**SHA256**消息身份验证码(MAC)的密码。

2. 重启**httpd**服务：

```
# systemctl restart httpd
```

**验证步骤**

1. 显示 Apache HTTP 服务器支持的密码列表：

   a. 安装**nmap**软件包：

   ```
   # dnf install nmap
   ```

   b. 使用**nmap**工具来显示支持的加密：

   ```
   # 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
   ...
   ```

**其它资源**

- **update-crypto-policies(8)** 手册页
- 使用系统范围的加密策略。
- SSLCipherSuite

## 1.9. 配置 TLS 客户端证书身份验证

客户端证书身份验证可让管理员只允许使用证书进行身份验证的用户访问 web 服务器上的资源。您可以为 **/var/www/html/Example/** 目录配置客户端证书身份验证。

如果 Apache HTTP 服务器使用 TLS 1.3 协议，某些客户端将需要额外的配置。例如，在 Firefox 中，将**about:config**菜单中的**security.tls.enable_post_handshake_auth**参数设置为**true**。详情请查看 Red Hat Enterprise Linux 8中的传输层安全版本1.3。

**先决条件**

- TLS 加密在服务器上启用，如 将 TLS 加密添加到 Apache HTTP 服务器 中所述。

**流程**

1. 编辑**/etc/httpd/conf/httpd.conf**文件，并将以下设置添加到你要为其配置客户端验证的**<VirtualHost>**指令中：

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

   **SSLVerifyClient require**设置定义了服务器必须成功验证客户端证书，然后客户端才能访问**/var/www/html/Example/**目录中的内容。

2. 重启**httpd**服务：

```
# systemctl restart httpd
```

**验证步骤**

1. 使用**curl**工具在没有客户端身份验证的情况下访问**https://example.com/Example/**URL：

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

这个错误表示 web 服务器需要客户端证书验证。

2. 将客户端私钥和证书以及 CA 证书传递给**curl**以便使用客户端身份验证来访问相同的URL：

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

如果请求成功，**curl**会显示存储在**/var/www/html/Example/**目录中的**index.html**文件。

**其它资源**

- mod_ssl 配置

## 1.10. 使用 MODSECURITY 在 WEB 服务器上保护 WEB 应用程序

ModSecurity 是一个开源 Web 应用程序防火墙(WAF)，受到各种 web 服务器（如 Apache、Nginx 和 IIS）支持，它可以降低 web 应用程序中的安全风险。ModSecurity 为配置服务器提供了可自定义的规则集。

**mod_security-crs** 软件包包含核心规则集(CRS)，针对跨 Web 站点脚本、错误的用户代理、SQL 注入、Trojans、会话劫持和其他利用的规则。

### 1.10.1. 为 Apache 部署基于 web 的 ModSecurity 应用程序防火墙

要通过部署 ModSecurity 来降低在 web 服务器上运行的基于 Web 的应用程序的风险，请为 Apache HTTP 服务器安装 **mod_security** 和 **mod_security_crs** 软件包。**mod_security_crs** 软件包为 ModSecurity 基于 Web 的应用程序防火墙(WAF)模块提供核心规则集(CRS)。

**流程**

1. 安装 **mod_security**、**mod_security_crs** 和 **httpd** 软件包：

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

2. 启动 **httpd** 服务器：

```
# systemctl restart httpd
```

**验证**

1. 验证 Apache HTTP 服务器上是否启用了 ModSecurity 基于 web 的应用程序防火墙：

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

2. 检查 **/etc/httpd/modsecurity.d/activated_rules/** 目录是否包含由 **mod_security_crs** 提供的规则：

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

**其它资源**

- Red Hat JBoss 核心服务 ModSecurity 指南
- Linux sysadmins 的 Web 应用程序防火墙简介

### 1.10.2. 向 ModSecurity 中添加自定义规则

如果 ModSecurity 核心规则集(CRS)中包含的规则不能适合您的场景，如果您想要阻止其他可能的攻击，则可以将自定义规则添加到 ModSecurity 基于 web 的应用防火墙使用的规则集中。以下示例演示了添加一条简单的规则。有关创建更复杂的规则，请参阅 ModSecurity Wiki 网站上的参考手册。

**先决条件**

- ModSecurity for Apache 已安装并启用。

**流程**

1. 在您选择的文本编辑器中打开 **/etc/httpd/conf.d/mod_security.conf** 文件，例如：

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

2. 在以 **SecRuleEngine On** 开头的行后添加以下示例规则：

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

   如果 **data** 参数包含 **evil** 字符串，则前面的规则禁止用户使用资源。

3. 保存更改，退出编辑器。

4. 重启 **httpd** 服务器：

```
# systemctl restart httpd
```

**验证**

1. 创建一个 ***test*.html** 页面：

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

2. 重启 **httpd** 服务器：

```
# systemctl restart httpd
```

3. 请求 ***test.html***，而在 HTTP 请求的 **GET** 变量没有恶意数据：

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

mod_security test
```

4. 请求 ***test.html***，而在 HTTP 请求的 **GET** 变量中带有恶意数据：

```
$ 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 do not have permission to access this resource.</p>
</body></html>
```

5. 检查 **/var/log/httpd/error_log** 文件，并找到带有 **param data containing an evil data** 信息的拒绝访问的日志条目：

```
[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"]
```

**其它资源**

- ModSecurity Wiki

## 1.11. 安装 APACHE HTTP 服务器手册

您可以安装 Apache HTTP 服务器手册。手册提供了详细信息，例如：

- 配置参数和指令
- 性能调整
- 身份验证设置
- 模块
- 内容缓存
- 安全提示
- 配置 TLS 加密

安装后，您可以使用 Web 浏览器显示手册。

**先决条件**

- Apache HTTP 服务器已安装并运行。

**流程**

1. 安装**httpd-manual**软件包：

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

2. 可选：默认情况下，所有连接到 Apache HTTP 服务器的客户端都可以显示手册。要限制对特定 IP 范围的访问，如**192.0.2.0/24** 子网，编辑**/etc/httpd/conf.d/manual.conf**文件，并将**Require ip 192.0.2.0/24**设置添加到 **<Directory "/usr/share/httpd/manual">**指令中：

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

3. 重启**httpd**服务：

```
# systemctl restart httpd
```

**验证步骤**

1. 要显示 Apache HTTP 服务器手册，使用 Web 浏览器连接到**http://*host_name_or_IP_address*/manual/**

## 1.12. 使用 APACHE 模块

**httpd** 服务是一个模块化应用程序，您可以使用多个 *Dynamic Shared Objects* (**DSO**s)对其进行扩展。*Dynamic Shared Objects* 是您可以可以根据需要在运行时动态加载或卸载的模块。您可以在 **/usr/lib64/httpd/modules/** 目录中找到这些模块。

### 1.12.1. 载入 DSO 模块

作为管理员，您可以通过配置服务器应加载哪个模块来选择要包含在服务器中的功能。若要载入特定的 DSO 模块，可使用**LoadModule** 指令。请注意，由单独的包提供的模块通常在**/etc/httpd/conf.modules.d/**目录中有自己的配置文件。

**先决条件**

- 您已安装了 **httpd** 软件包。

**流程**

1. 在 **/etc/httpd/conf.modules.d/** 目录中的配置文件中搜索模块名称：

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

2. 编辑在其中找到了模块名称的配置文件，然后取消对模块的 **LoadModule** 指令的注释：

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

3. 如果没有找到模块，例如，因为 RHEL 软件包没有提供该模块，请创建一个配置文件，如 **/etc/httpd/conf.modules.d/30-example.conf** ：

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

4. 重启**httpd**服务：

```
# systemctl restart httpd
```

### 1.12.2. 编译自定义 Apache 模块

您可以创建自己的模块，并使用 **httpd-devel** 软件包帮助构建，该软件包包含 include 文件、头文件以及编译模块所需的 **APache eXtenSion** (**pxs**)工具。

**先决条件**

- 您已安装了 **httpd-devel** 软件包。

**流程**

- 使用以下命令构建自定义模块：

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

**验证步骤**

- 加载模块的方式与 加载 DSO 模块 中所述的方法一样。

## 1.13. 从 NSS 数据库导出私钥和证书，以便在 APACHE WEB 服务器配置中使用它们

因为 RHEL 8 不再为 Apache web 服务器提供 **mod_nss** 模块，因此红帽建议使用 **mod_ssl** 模块。如果您将私钥和证书存储在网络安全服务(NSS)数据库中，请按照以下步骤以 Privacy Enhanced 邮件(PEM)格式提取密钥和证书。

## 1.14. 其他资源

- **httpd(8)** man page
- **httpd.service(8)** man page
- **httpd.conf(5)** man page
- **apachectl(8)** man page
- Apache HTTP 服务器上的 Kerberos 身份验证： 对 Apache httpd 操作使用 GSS-Proxy。使用 Kerberos 是在 Apache HTTP 服务器中强制进行客户端授权的替代方法。
- 将应用程序配置为通过 PKCS #11 使用加密硬件。

# 第 2 章 设置和配置 NGINX

NGINX 是一个高性能和模块化的服务器，可作为：

- Web 服务器
- 反向代理服务器
- 负载均衡器

这部分论述了如何在这些场景中使用 NGINX。

## 2.1. 安装并准备 NGINX

在 Red Hat Enterprise Linux 9 中，Application Streams 提供了不同版本的 NGINX 。通过使用默认配置，NGINX 在端口 **80** 上作为 Web 服务器运行，并提供 **/usr/share/nginx/html/** 目录中的内容。

**先决条件**

- 已安装 RHEL 9。
- 主机订阅了红帽客户门户网站。
- **firewalld**服务已经启用并启动。

**流程**

1. 安装**nginx**软件包：
   - 要将 NGINX 1.20 作为 RPM 软件包的此应用程序流的初始版本安装：

     ```
     # dnf install nginx
     ```

     **注意**

     如果您之前启用了 NGINX 模块流，这个命令会从启用的流中安装 NGINX 版本。

   - 要从模块流安装替代的 NGINX 后期版本：

     a. 显示可用的 NGINX 模块流：

     ```
     # dnf module list nginx
     ...
     rhel-AppStream
     Name       Stream       Profiles       Summary
     nginx      1.22         common [d]     nginx webserver
     ...
     Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
     ```

     b. 启用所选流：

     ```
     # dnf module enable nginx:stream_version
     ```

c. 安装 nginx 软件包：

```
# dnf install nginx
```

2. 打开 NGINX 应该在其防火墙中提供其服务的端口。例如，要在**firewalld**中为HTTP（端口 80）和 HTTPS（端口 443）开放默认端口，请输入：

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

3. 设置**nginx**服务在系统启动时自动启动:

```
# systemctl enable nginx
```

4. 另外，也可启动**nginx**服务：

```
# systemctl start nginx
```

如果您不想使用默认配置，请跳过这一步，并在启动该服务前相应地配置 NGINX。

**验证步骤**

1. 使用 **dnf** 工具验证是否已安装 **nginx** 软件包。

- 如果是 NGINX 1.20 RPM 软件包：

```
# dnf list installed nginx
Installed Packages
nginx.x86_64    1:1.20.1-4.el9      @rhel-AppStream
```

- 如果是所选的是 NGINX 模块流：

```
# dnf list installed nginx
Installed Packages
nginx.x86_64    1:1.22.1-3.module+el9.2.0+17617+2f289c6c    @rhel-AppStream
```

2. 确保在 firewalld 中打开了 NGINX 需要的端口：

```
# firewall-cmd --list-ports
80/tcp 443/tcp
```

3. 验证**nginx**服务是否已启用：

```
# systemctl is-enabled nginx
enabled
```

**其他资源**

- 使用和配置 Subscription Manager
- 安全网络

## 2.2. 将 NGINX 配置为一个为不同域提供不同内容的 WEB 服务器

默认情况下，NGINX 作为 web 服务器，为与服务器的 IP 地址关联的所有域名提供相同的内容。此流程解释了如何配置 NGINX 来实现一下情况：

- 使用 **/var/www/example.com/** 目录中的内容提供对 **example.com** 域的请求
- 使用**/var/www/example.net/**目录中的内容为**example.net**域提供请求
- 使用**/usr/share/nginx/html/**目录中的内容为所有其他请求提供服务，例如，向服务器的IP地址或与服务器的IP地址相关联的其他域发送请求

**先决条件**

- NGINX 已安装
- 客户端和 Web 服务器将 **example.com** 和 **example.net** 域解析为 Web 服务器的 IP 地址。请注意，您必须手动将这些条目添加到 DNS 服务器中。

**流程**

1. 编辑**/etc/nginx/nginx.conf**文件：
   a. 默认情况下，**/etc/nginx/nginx.conf**文件已包含catch-all配置。如果您已从配置中删除了这部分，请将以下**server**块重新添加到**/etc/nginx/nginx.conf**文件中的**http**块中：

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

   这些设置配置以下内容：

   - **listen**指令定义服务监听的 IP 地址和端口。在本例中，NGINX 监听所有 IPv4 和 IPv6 地址的**80**端口 。**default_server**参数表示，NGINX 使用此**server**块作为匹配 IP 地址和端口的请求的默认值。
   - **server_name**参数定义此**server**块所负责的主机名。将**server_name**设置为**_**会将NGINX配置为接受这个**server**块的任何主机名。
   - **root**指令设置此**server**块的 Web 内容的路径。

   b. 将类似于**example.com**域的**server**块添加到 **http**块中：

```
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**指令为此域定义一个单独的访问日志文件。
   - **error_log**指令为此域定义单独的错误日志文件。

c. 将类似于 **example.com** 域的一个 **server** 块添加到 **http** 块中：

```
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. 为这两个域创建根目录：

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

3. 在两个根目录中设置**httpd_sys_content_t**上下文：

```
# 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/
```

这些命令在**/var/www/example.com/**和**/var/www/example.net/**目录中设置**httpd_sys_content_t**上下文。

请注意，您必须安装 **policycoreutils-python-utils**软件包才能运行**restorecon**命令。

4. 为这两个域创建日志目录：

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

5. 重启**nginx**服务：

```
# systemctl restart nginx
```

**验证步骤**

1. 在每个虚拟主机的文档 root 中创建不同的示例文件：

```
# 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. 使用浏览器并连接到 **http://example.com**Web 服务器显示**/var/www/example.com/index.html**文件中的示例内容。

3. 使用浏览器并连接到 **http://example.net**Web 服务器显示**/var/www/example.net/index.html**文件中的示例内容。

4. 使用浏览器连接到**http://*IP_address_of_the_server***。Web 服务器显示**/usr/share/nginx/html/index.html**文件中的示例内容。

## 2.3. 在 NGINX WEB 服务器中添加 TLS 加密

您可以在 NGINX web 服务器上为**example.com** 域启用 TLS 加密。

**先决条件**

- NGINX 已安装。
- 私钥存储在 **/etc/pki/tls/private/example.com.key** 文件中。
  有关创建私钥和证书签名请求(CSR)的详细信息，以及如何从证书颁发机构(CA)请求证书，请参阅您的 CA 文档。
- TLS 证书存储在**/etc/pki/tls/certs/example.com.crt**文件中。如果您使用其他路径，请调整该流程的对应步骤。
- CA 证书已附加到服务器的 TLS 证书文件中。
- 客户端和网页服务器会将服务器的主机名解析为 web 服务器的 IP 地址。
- 在本地防火墙中打开端口**443**。
- 如果服务器运行 RHEL 9.2 或更高版本，并且启用了 FIPS 模式，则客户端必须支持 Extended Master Secret(EMS)扩展或使用 TLS 1.3。没有 EMS 的 TLS 1.2 连接会失败。如需更多信息，请参阅 强制 TLS 扩展"Extended Master Secret" 知识库文章。

**流程**

1. 编辑 **/etc/nginx/nginx.conf**文件，并将以下**server**块添加到配置中的**http**块中：

```
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. 可选：从 RHEL 9.3 开始，您可以使用 **ssl_pass_phrase_dialog** 指令为每个加密私钥配置一个在 **nginx** 启动时调用的外部程序。向 **/etc/nginx/nginx.conf** 文件中添加以下行：
   - 要为每个加密的私钥文件调用外部程序，请输入：

     ```
     ssl_pass_phrase_dialog exec:<path_to_program>;
     ```

     NGINX 使用以下两个参数调用该程序：
     - 服务器名称在 **server_name** 设置中指定。
     - 以下一种算法之一：**RSA**、**DSA**、**EC**、**DH** 或 **UNK**（如果无法识别加密算法）。
   - 如果要为每个加密的私钥文件手动输入密码短语，请输入：

     ```
     ssl_pass_phrase_dialog builtin;
     ```

     如果没有配置 **ssl_pass_phrase_dialog**，这是默认行为。

**注意**

如果使用此方法，但至少有一个私钥受密码保护，则 **nginx** 服务无法启动。在这种情况下，请使用其它方法。

- 如果您希望 **systemd** 在使用 **systemctl** 工具启动 **nginx** 服务时对每个加密的私钥提示输入密码短语，请输入：

```
ssl_pass_phrase_dialog exec:/usr/libexec/nginx-ssl-pass-dialog;
```

3. 出于安全考虑，配置成只有 **root** 用户才可以访问私钥文件：

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

**警告**

如果私钥被设置为可以被未授权的用户访问，则需要撤销证书，然后再创建一个新私钥并请求一个新证书。否则，TLS 连接就不再安全。

4. 重启**nginx**服务：

```
# systemctl restart nginx
```

**验证步骤**

- 使用浏览器连接到**https://example.com**

**其它资源**

- RHEL 中 TLS 的安全性注意事项

## 2.4. 将 NGINX 配置为 HTTP 流量的反向代理

您可以将 NGINX web 服务器配置为作为 HTTP 流量的反向代理。例如，您可以使用此功能将请求转发到远程服务器上的特定子目录。从客户端的角度来看，客户端从它所访问的主机加载内容。但是 NGINX 会从远程服务器加载实际内容并将其转发给客户端。

这个流程解释了如何将流向web 服务器上的**/example**目录的流量转发到URL**https://example.com**。

**先决条件**

- NGINX 已安装，如 安装和准备 NGINX 中所述。
- 可选：反向代理上启用了 TLS 加密。

**流程**

1. 编辑**/etc/nginx/nginx.conf**文件，并将以下设置添加到提供反向代理的**server**块中：

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

**location**块定义了 NGINX 将**/example**目录中的所有请求传给**https://example.com**。

2. 将**httpd_can_network_connect**SELinux 布尔值参数设置为**1**，以便将 SELinux 设置为允许 NGINX 转发流量：

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

3. 重启**nginx**服务：

```
# systemctl restart nginx
```

**验证步骤**

- 使用浏览器连接到 **http://*host_name*/example**，就会显示**https://example.com**的内容。

## 2.5. 将 NGINX 配置为 HTTP 负载均衡器

您可以使用 NGINX 反向代理功能进行负载均衡流量。这个步骤描述了如何将 NGINX 配置为 HTTP 负载均衡器。它会根据服务器上的活跃连接的数量，将请求发送到不同服务器（发送到活跃连接数量最小的服务器）。如果两个服务器都不可用，这个过程还定义了第三个主机用于回退。

**先决条件**

- NGINX 已安装，如 安装和准备 NGINX 中所述。

**流程**

1. 编辑**/etc/nginx/nginx.conf**文件并添加以下设置：

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

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

在名为**backend**的主机组中的**least_conn**指令定义了 NGINX 将请求发送到**server1.example.com**或**server2.example.com**，具体取决于哪个主机具有最少的活动连接数。NGINX 仅在其他两个主机不可用时使用**server3.example.com**作为备份。

**proxy_pass**指令设置为**http://backend** 时，NGINX 充当反向代理，并使用**backend**主机组根据该组的设置分发请求。

您还可以指定其他方法，而不是**least_conn**负载均衡方法：

- 不指定方法，使用轮询的方式在服务器间平均分发请求。
- **ip_hash**根据从 IPv4 地址的前三个八位字节或客户端的整个 IPv6 地址计算的哈希值将来自一个客户端地址的请求发送到同一台服务器。
- **hash**，根据用户定义的密钥（可以是字符串、变量或两者的组合）来确定服务器。用**consistent**参数来进行配置，NGINX可根据用户定义的哈希密钥值向所有的服务器分发请求。
- **random**将请求发送到随机挑选的服务器。

2. 重启**nginx**服务：

```
# systemctl restart nginx
```

## 2.6. 其他资源

- 官方 NGINX 文档.请注意，红帽并不维护这个文档，并且可能无法与您安装的 NGINX 版本一起使用。
- 将应用程序配置为通过 PKCS #11 使用加密硬件。

# 第 3 章 配置 SQUID 缓存代理服务器

Squid 是一个代理服务器，可缓存内容以减少带宽并更快地加载 Web 页面。本章论述了如何将 Squid 设置为 HTTP、HTTPS 和 FTP 协议的代理，以及验证和限制访问。

## 3.1. 将 SQUID 设置为没有身份验证的缓存代理

您可以将 Squid 配置为没有身份验证的缓存代理。此流程会根据 IP 范围限制对代理的访问。

**先决条件**

- 流程假设 **/etc/squid/squid.conf** 文件是由 **squid** 软件包提供的。如果您在之前编辑了这个文件，请删除该文件并重新安装该软件包。

**流程**

1. 安装 **squid** 软件包：

```
# dnf install squid
```

2. 编辑 **/etc/squid/squid.conf** 文件：

   a. 调整 **localnet** 访问控制列表(ACL)以匹配应该允许使用代理的 IP 范围：

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

   默认情况下，**/etc/squid/squid.conf** 文件包含 **http_access allow localnet** 规则，该规则允许使用 **localnet** ACL 中指定的所有 IP 范围的代理。请注意，您必须在 **http_access allow localnet** 规则前指定所有 **localnet** ACL。

**重要**

删除所有与您的环境不匹配的现有 **acl localnet** 条目。

   b. 以下 ACL 存在于默认配置中，并将 **443** 定义为使用 HTTPS 协议的端口：

```
acl SSL_ports port 443
```

   如果用户也可以在其它端口上使用 HTTPS 协议，请为每个端口添加 ACL：

```
acl SSL_ports port port_number
```

   c. 更新 **acl Safe_ports** 规则列表，以配置 Squid 可对哪个端口建立连接。例如，若要配置使用代理的客户端只能访问端口 21(FTP)、80(HTTP)和 443(HTTPS)上的资源，请在配置中只保留以下 **acl Safe_ports** 语句：

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

默认情况下，配置包含 **http_access deny !Safe_ports** 规则，该规则定义禁止对定义在 **Safe_ports** ACL 中端口的访问。

d. 在 **cache_dir** 参数中配置缓存类型、缓存目录的路径、缓存大小以及特定于其它缓存类型的设置：

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

有了这些设置：

- squid 使用 **ufs** 缓存类型。
- Squid 将其缓存存储在 **/var/spool/squid/** 目录中。
- 缓存增长到 **10000** MB。
- Squid 在 **/var/spool/squid/** 目录中创建 **16** level-1 子目录。
- Squid 在每个 level-1 目录中创建 **256** 个子目录。
  如果您没有设置 **cache_dir** 指令，Squid 会在内存中存储缓存。

3. 如果您在 **cache_dir** 参数中设置了与 **/var/spool/squid/** 不同的缓存目录：

a. 创建缓存目录：

```
# mkdir -p path_to_cache_directory
```

b. 配置缓存目录的权限：

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

c. 如果您在 **enforcing** 模式下运行 SELinux，请为缓存目录设置 **squid_cache_t** 上下文：

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

如果系统上没有 **semanage** 工具，请安装 **policycoreutils-python-utils** 软件包。

4. 在防火墙中打开 **3128** 端口：

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

5. 启用并启动 **squid** 服务：

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

**验证步骤**

要验证代理是否正常工作，请使用 **curl** 工具下载网页：

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

如果 **curl** 没有显示任何错误，并且 **index.html** 文件可以下载到当前目录中，那么代理工作正常。

## 3.2. 使用 LDAP 身份验证将 SQUID 设置为缓存代理

您可以将 Squid 配置为使用 LDAP 验证用户身份的缓存代理。此流程配置仅经过身份验证的用户可以使用代理。

**先决条件**

- 流程假设 **/etc/squid/squid.conf** 文件是由 **squid** 软件包提供的。如果您在之前编辑了这个文件，请删除该文件并重新安装该软件包。
- LDAP 目录中存在一个服务用户，如 **uid=proxy_user,cn=users,cn=accounts,dc=example,dc=com**。Squid 只使用此帐户搜索验证用户。如果存在身份验证用户，Squid 会以此用户的身份绑定到该目录以验证身份验证。

**流程**

1. 安装 **squid** 软件包：

```
# dnf install squid
```

2. 编辑 **/etc/squid/squid.conf** 文件：

   a. 要配置 **basic_ldap_auth** 帮助工具，请在 **/etc/squid/squid.conf** 的顶部添加以下配置条目：

```
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
```

   下面描述了在上面示例中传给 **basic_ldap_auth** 帮助工具的参数：

   - **-b *base_DN*** 设置 LDAP 搜索基础。
   - **-D *proxy_service_user_DN*** 设置 Squid 用来搜索目录中验证用户的帐户的可分辨名称 (DN)。
   - **-W *path_to_password_file*** 设置包含代理服务用户密码的文件路径。使用密码文件可防止在操作系统的进程列表中看到密码。
   - **-f *LDAP_filter*** 指定 LDAP 搜索过滤器。Squid 将 **%s** 变量替换为验证用户提供的用户名。
     示例中的 **(&(objectClass=person)(uid=%s))** 过滤器定义用户名必须与 **uid** 属性中设置的值匹配，并且与包含 **person** 对象类的目录条目匹配。
   - **-ZZ** 使用 **STARTTLS** 命令强制实现通过 LDAP 协议的 TLS 加密连接。在以下情况下省略 **-ZZ** ：
     - LDAP 服务器不支持加密的连接。
     - URL 中指定的端口使用 LDAPS 协议。
   - -H LDAP_URL 参数指定协议、主机名或 IP 地址以及 LDAP 服务器的端口，格式为 URL。

b. 添加以下 ACL 和规则来配置 Squid 只允许经过身份验证的用户使用代理：

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

**重要**

在 **http_access deny** 所有规则前指定这些设置。

c. 删除以下规则以禁止绕过 **localnet** ACL 中指定的 IP 范围的代理身份验证：

```
http_access allow localnet
```

d. 以下 ACL 存在于默认配置中，并将 **443** 定义为使用 HTTPS 协议的端口：

```
acl SSL_ports port 443
```

如果用户也可以在其它端口上使用 HTTPS 协议，请为每个端口添加 ACL：

```
acl SSL_ports port port_number
```

e. 更新 **acl Safe_ports** 规则列表，以配置 Squid 可对哪个端口建立连接。例如，若要配置使用代理的客户端只能访问端口 21(FTP)、80(HTTP)和 443(HTTPS)上的资源，请在配置中只保留以下 **acl Safe_ports** 语句：

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

默认情况下，配置包含 **http_access deny !Safe_ports** 规则，该规则定义拒绝访问 **Safe_ports ACL** 中未定义的端口。

f. 在 **cache_dir** 参数中配置缓存类型、缓存目录的路径、缓存大小以及特定于其它缓存类型的设置：

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

有了这些设置：

- squid 使用 **ufs** 缓存类型。
- Squid 将其缓存存储在 **/var/spool/squid/** 目录中。
- 缓存增长到 **10000** MB。
- Squid 在 **/var/spool/squid/** 目录中创建 **16** level-1 子目录。
- Squid 在每个 level-1 目录中创建 **256** 个子目录。
  如果您没有设置 **cache_dir** 指令，Squid 会在内存中存储缓存。

3. 如果您在 **cache_dir** 参数中设置了与 **/var/spool/squid/** 不同的缓存目录：

a. 创建缓存目录：

```
# mkdir -p path_to_cache_directory
```

b. 配置缓存目录的权限：

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

c. 如果您在 **enforcing** 模式下运行 SELinux，请为缓存目录设置 **squid_cache_t** 上下文：

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

如果系统上没有 **semanage** 工具，请安装 **policycoreutils-python-utils** 软件包。

4. 将 LDAP 服务用户的密码存储在 **/etc/squid/ldap_password** 文件中，并为该文件设置合适的权限：

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

5. 在防火墙中打开 **3128** 端口：

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

6. 启用并启动 **squid** 服务：

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

**验证步骤**

要验证代理是否正常工作，请使用 **curl** 工具下载网页：

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

如果 curl 没有显示任何错误，并且 **index.html** 文件已下载到当前目录中，则代理工作正常。

**故障排除步骤**

验证 helper 工具是否正常工作：

1. 使用您在 **auth_param** 参数中使用的相同设置手动启动 helper 工具：

```
# /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. 输入一个有效的用户名和密码，然后按 **Enter** 键：

```
user_name password
```

如果 helper 工具返回 **OK**，则身份验证成功。

## 3.3. 将 SQUID 设置为带有 KERBEROS 身份验证的缓存代理

您可以将 Squid 配置为使用 Kerberos 向活动目录(AD)验证用户的缓存代理。此流程配置仅经过身份验证的用户可以使用代理。

**先决条件**

- 流程假设 **/etc/squid/squid.conf** 文件是由 **squid** 软件包提供的。如果您在之前编辑了这个文件，请删除该文件并重新安装该软件包。
- 要安装 Squid 的服务器是 AD 域的成员。

**流程**

1. 安装以下软件包：

```
# dnf install squid krb5-workstation
```

2. 以 AD 域管理员身份进行身份验证：

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

3. 为 Squid 创建 keytab，并将其存储在 **/etc/squid/HTTP.keytab** 文件中：

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

4. 向 keytab 添加 **HTTP** 服务主体：

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

5. 将 keytab 文件的拥有者设为 **squid** 用户：

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

6. （可选）验证 keytab 文件包含代理服务器的完全限定域名(FQDN)的 **HTTP** 服务主体：

```
# 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. 编辑 **/etc/squid/squid.conf** 文件：

   a. 要配置 **negotiate_kerberos_auth** 帮助工具，请将以下配置条目添加到 **/etc/squid/squid.conf** 的顶部：

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

下面描述了在上例中传给 **negotiate_kerberos_auth** 帮助工具的参数：

- **-k *file*** 设置 keytab 文件的路径。请注意，squid 用户必须拥有这个文件的读取权限。
- **-s HTTP/*host_name@kerberos_realm*** 设置 Squid 使用的 Kerberos 主体。

另外，您可以通过将以下一个或多个参数传给帮助工具来启用日志：

- **-i** 记录信息，如验证用户。
- **-d** 启用调试日志记录。
  Squid 将帮助工具中的调试信息记录到 **/var/log/squid/cache.log** 文件。

b. 添加以下 ACL 和规则来配置 Squid 只允许经过身份验证的用户使用代理：

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

**重要**

在 **http_access deny all** 规则前指定这些设置。

c. 删除以下规则以禁止绕过 **localnet** ACL 中指定的 IP 范围的代理身份验证：

```
http_access allow localnet
```

d. 以下 ACL 存在于默认配置中，并将 **443** 定义为使用 HTTPS 协议的端口：

```
acl SSL_ports port 443
```

如果用户也可以在其它端口上使用 HTTPS 协议，请为每个端口添加 ACL：

```
acl SSL_ports port port_number
```

e. 更新 **acl Safe_ports** 规则列表，以配置 Squid 可对哪个端口建立连接。例如，若要配置使用代理的客户端只能访问端口 21(FTP)、80(HTTP)和 443(HTTPS)上的资源，请在配置中只保留以下 **acl Safe_ports** 语句：

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

默认情况下，配置包含 **http_access deny !Safe_ports** 规则，该规则定义禁止对定义在 **Safe_ports** ACL 中端口的访问。

f. 在 **cache_dir** 参数中配置缓存类型、缓存目录的路径、缓存大小以及特定于其它缓存类型的设置：

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

有了这些设置：

- squid 使用 **ufs** 缓存类型。

- Squid 将其缓存存储在 **/var/spool/squid/** 目录中。
- 缓存增长到 **10000** MB。
- Squid 在 **/var/spool/squid/** 目录中创建 **16** level-1 子目录。
- Squid 在每个 level-1 目录中创建 **256** 个子目录。
  如果您没有设置 **cache_dir** 指令，Squid 会在内存中存储缓存。

8. 如果您在 **cache_dir** 参数中设置了与 **/var/spool/squid/** 不同的缓存目录：

   a. 创建缓存目录：

   ```
   # mkdir -p path_to_cache_directory
   ```

   b. 配置缓存目录的权限：

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

   c. 如果您在 **enforcing** 模式下运行 SELinux，请为缓存目录设置 **squid_cache_t** 上下文：

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

   如果系统上没有 **semanage** 工具，请安装 **policycoreutils-python-utils** 软件包。

9. 在防火墙中打开 **3128** 端口：

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

10. 启用并启动 **squid** 服务：

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

**验证步骤**

要验证代理是否正常工作，请使用 **curl** 工具下载网页：

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

如果 **curl** 没有显示任何错误，并且当前目录中存在 **index.html** 文件，则代理工作正常。

**故障排除步骤**

手动测试 Kerberos 身份验证：

1. 为 AD 帐户获取 Kerberos ticket：

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

2. 显示 ticket（可选）：

```
# klist
```

3. 使用 **negotiate_kerberos_auth_test** 工具测试身份验证：

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

如果助手工具返回令牌，则身份验证成功：

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

## 3.4. 在 SQUID 中配置域拒绝列表

通常,管理员想要阻止对特定域的访问。这部分论述了如何在 Squid 中配置域拒绝列表。

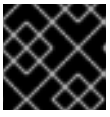
**先决条件**

- squid 被配置，用户可以使用代理。

**流程**

1. 编辑 **/etc/squid/squid.conf** 文件，并添加以下设置：

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

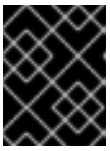


**重要**

在允许访问用户或客户端的第一个 **http_access allow** 语句前添加这些条目。

2. 创建 **/etc/squid/domain_deny_list.txt** 文件，并添加您要阻止的域。例如，要阻止对 **example.com**（包括子域）以及对 **example.net** 的访问，请添加：

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



**重要**

如果您引用了 squid 配置中的 **/etc/squid/domain_deny_list.txt** 文件，则该文件不能为空。如果文件为空，Squid 无法启动。

3. 重启 **squid** 服务：

```
# systemctl restart squid
```

## 3.5. 将 SQUID 服务配置为监听特定端口或 IP 地址

默认情况下，Squid 代理服务侦听所有网络接口上的 **3128** 端口。您可以更改端口，并将 Squid 配置为监听特定的 IP 地址。

**先决条件**

- **squid** 软件包已安装。

**流程**

1. 编辑 **/etc/squid/squid.conf** 文件：
   - 要设置 Squid 服务侦听的端口，请在 **http_port** 参数中设置端口号。例如，要将端口设为 **8080**，请设置：

     ```
     http_port 8080
     ```

   - 要配置 Squid 服务侦听的 IP 地址，请在 **http_port** 参数中设置 IP 地址和端口号。例如，若要配置 Squid 仅侦听**192.0.2.1** IP 地址的端口 **3128**，请设置：

     ```
     http_port 192.0.2.1:3128
     ```

     向配置文件中添加多个 **http_port** 参数，以配置 Squid 侦听多个端口和 IP 地址：

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

2. 如果您配置了 Squid 使用不同的端口作为默认值(**3128**)：
   a. 在防火墙中打开端口：

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

   b. 如果您在 enforcing 模式下运行 SELinux，请将端口分配给 **squid_port_t** 端口类型定义：

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

      如果系统上没有 **semanage** 工具，请安装 **policycoreutils-python-utils** 软件包。

3. 重启 **squid** 服务：

   ```
   # systemctl restart squid
   ```

## 3.6. 其它资源

- 配置参数 **usr/share/doc/squid-<version>/squid.conf.documented**