1. Zawartość dokumentu

Celem dokumentu jest przedstawienie sposobu instalacji oraz konfiguracji serwera WWW Apache2 w systemie OpenBSD-3.6 z naciskiem na bezpieczeństwo takiego rozwiązania. Adresowany jest on do administratora systemu, zaś opisywane czynności powinny być wykonywane po skonfigurowaniu wszystkich ustawień i zabezpieczeń protokołów TCP/IP.

2. Materiały źródłowe

Wykorzystywane materiały źródłowe:

- Setting Up a Secure Apache 2 Server,
  http://www.samspublishing.com/articles/article.asp?p=30115

  Dokument zawiera opis konfiguracji protokołu SSL w Apache2, niestety nie zawiera nawet odnośników do żadnych innych kwestii związanych z bezpieczeństwem (co przeczy tytułowi).

- Securing Apache 2: Step-by-Step,
  http://www.securityfocus.com/infocus/1786

  Dokument opisuje bezpieczną instalacje oraz konfiguracje Apache2 z wykorzystaniem mechanizmu chroot(8). Jednym z założeń jest obsługiwanie jedynie statycznych serwisów WWW, co może tłumaczyć pominięcie konfiguracji SSL w procesie konfiguracji.

- Setting up a Secure HTTP server with SSL(8),
  http://www.openbsd.org/faq/faq10.html#HTTPS

  Bardzo krótka instrukcja autorów OpenBSD opisująca konfigurację SSL w domyślnej instalacji systemu, która zawiera jednak oprogramowanie Apache bazujące na starszej wersji 1.3.29.

- Locking Down Apache,
  http://www.bastille-linux.org/jay/Talks/slides-defcon-securing-apache.pdf

  Najpełniejszy i najbardziej obszerny opis zabezpieczania Apache, zawiera opis wyłączenia obsługi metody `TRACE`. Żadnych wzmianek o konfiguracji SSL.

3. Wprowadzenie – dlaczego Apache2

Apache jest obecnie zdecydowanie najbardziej popularnym wśród serwisów WWW (wg Netcraft) obsługuje juz 67% domen. Powszechnie uważany jest za oprogramowanie bezpieczne, a ogromna liczba użytkowników zapewnia doskonałe wsparcie. W połączeniu z cieszącymi się ogromną popularnością systemami Linux/*BSD stanowi

doskonałą darmową platformę dla hostingu serwisów WWW, nawet w zastosowaniach komercyjnych.

W celu zapewnienia maksymalnego bezpieczeństwa takiego rozwiązania należy zadbać o kompletność całego środowiska serwisu – od wyboru i instalacji systemu operacyjnego do konfiguracji serwera WWW. Wybierając system OpenBSD dostajemy od razu działający serwer Apache wraz z zabezpieczeniami SSL, powstaje więc pytanie, dlaczego z niego nie skorzystać?

Na skutek zmiany polityki licencyjnej Apache, która zabroniła stosowania tej "kuszącej" nazwy (przez komercyjne produkty, takie jak Stronghold) nie jest możliwe wprowadzanie poprawek i dalsze używanie nazwy Apache. Takiej zmiany nie mogło zaakceptować środowisko OpenBSD, znane z łatania i poprawiania zabezpieczeń w wielu projektach.

Apache2 oferuje jednak korzyści, które w określonych sytuacjach mogą przeważyć nad zaletami wyjątkowo ostrożnej polityki bezpieczeństwa OpenBSD, są to zwłaszcza:

- wykorzystanie wątków POSIX, co dodatkowo poprawia wydajność na platformach wieloprocesorowych,
- obsługa IPv6 – do tej pory konieczne było nakładanie odpowiednich łat na podstawową dystrybucję,
- zintegrowana obsługa SSL – do tej pory funkcja dostępna tylko poprzez wykorzystanie modułu mod_ssl lub (powstałego na jego bazie) Apache-SSL.

4. Instalacja

Z wcześniej opisanych powodów OpenBSD nie wspiera Apache2 ani przez umieszczenie go w bazowej dystrybucji, ani nawet w systemie portów. Inne systemy stosują odmienną politykę, jak np. Debian, który umieszcza pakiet binariów, jednak tylko dla dystrybucji testing oraz unstable. Podobnie FreeBSD przygotowało port o nazwie apache2.

Przed przystąpieniem do stworzenia własnej instalacji, należy określić założenia, co do funkcjonalności, jakiej wymagamy od Apache2 – będą to tylko bazowe funkcje, co ma ułatwić zabezpieczanie serwera. Listę z pozycji [2] poszerzamy o obsługę protokołu HTTPS, czyli będzie to:

- obsługa jedynie statycznych serwisów (bez CGI),
- obsługa domen wirtualnych,
- zapisywanie historii dostępu do stron (oraz ewentualnych błędów) do plików logów,
- kontrola dostępu na podstawie IP klienta oraz podstawowa autoryzacja użytkownik/hasło,
- ochrona przesyłanych danych z wykorzystaniem mechanizmów SSL,
- dodatkowe zabezpieczenia (wyłączenie metody `TRACE`).

Instalację należy rozpocząć od pobrania źródeł oraz zweryfikowania ich wiarygodności. Postępujemy tutaj zgodnie ze wskazówkami zamieszczonymi przez autorów Apache na

stronie http://httpd.apache.org/download.cgi – korzystniejsze może do tych czynności okazać się użycie innej od docelowej maszyny, jeżeli nie chcemy instalować zbędnych w późniejszej pracy pakietów. Możemy w takim przypadku jedynie skopiować pobrane, zweryfikowane źródła wykorzystując np scp(1). Na chwilę obecną aktualne wydanie Apache2 oznaczone jest numerem wersji 2.0.52.

W systemie OpenBSD do stworzenia konta użytkownika, na którym będzie działał Apache wykorzystujemy polecenia:

```
# useradd -c "Apache Server" -d /var/empty -g apache -s /sbin/nologin apache
# groupadd apache
```

Po rozpakowaniu kolej na konfiguracje – zgodnie z zasadą poprawy stopnia zabezpieczeń przez wyłączenie zbędnych funkcjonalności wskazujemy tylko rzeczywiście konieczne moduły. Za punkt wyjścia do stworzenia listy potrzebnych modułów można przyjąć opis konfiguracji podany w pozycji [2]. Czynności nie wymagające uprawnień administratora należy wykonywać z poziomu zwykłego użytkownika. Polecenie konfiguracji zawiera listę modułów, które nie będą wykorzystywane oraz dodatkowo włączenie obsługi SSL:

```
$ ./configure \
--prefix=/usr/local/apache2 \
--with-mpm=prefork \
--enable-ssl \
--enable-rewrite \
--disable-charset-lite \
--disable-include \
--disable-env \
--disable-setenvif \
--disable-status \
--disable-autoindex \
--disable-asis \
--disable-cgi \
--disable-negotiation \
--disable-imap \
--disable-actions \
--disable-userdir \
--disable-alias \
--disable-so
```

Następnie należy skompilować oraz zainstalować pakiet:

```
$ make
$ su
# make install
```

Weryfikacji listy włączonych modułów możemy dokonać poleceniem:

```
$ /usr/local/apache2/bin/httpd -l
Compiled in modules:
  core.c
  mod_access.c
  mod_auth.c
  mod_log_config.c
  mod_ssl.c
  prefork.c
  http_core.c
  mod_mime.c
  mod_dir.c
  mod_rewrite.c
```

Na potrzeby sprawdzenia instalacji stwórzmy dwie wirtualne domeny:

- *www.bss.lab* – dostęp tylko przez HTTP,

- *www.openbsd.lab* – dostęp tylko przez HTTPS.

Dla obsługi HTTPS potrzebujemy parę klucz–certyfikat SSL. Tworzymy je zgodnie z dokumentacją [3]:

```
# openssl genrsa -out /etc/ssl/private/www.openbsd.lab.key 1024
Generating RSA private key, 1024 bit long modulus
.....................................................++++++
.....................++++++
e is 65537 (0x10001)
#openssl req -new -key /etc/ssl/private/www.openbsd.lab.key \
   -out /etc/ssl/private/www.openbsd.lab.csr
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) []:PL
State or Province Name (full name) []:mazowieckie
Locality Name (eg, city) []:Warszawa
Organization Name (eg, company) []:Politechnika Warszawska, EiTI
Organizational Unit Name (eg, section) []:IAiIS
Common Name (eg, fully qualified host name) []:www.openbsd.lab
Email Address []:P.Trojanek@elka.pw.edu.pl

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
# openssl x509 -req -days 365 -in /etc/ssl/private/www.openbsd.lab.csr \
    -signkey /etc/ssl/private/www.openbsd.lab.key \
    -out /etc/ssl/www.openbsd.lab.crt
Signature ok
subject=/C=PL/ST=mazowieckie/L=Warszawa/O=Politechnika Warszawska,
EiTI/OU=IAiIS/CN=www.openbsd.lab/emailAddress=P.Trojanek@elka.pw.edu.pl
Getting Private key
```

Po utworzeniu katalogów z zawartością serwisów domen możemy przetestować instalację. W tym celu należy stworzyć odpowiedni plik konfiguracyjny (patrz dodatek A). Sprawdzamy poprawność pliku konfiguracyjnego, a następnie uruchamiamy demona:

```
# /usr/local/apache2/bin/apachectl -t
Syntax OK
# /usr/local/apache2/bin/apachectl -k start
```

Już w tym momencie możemy przetestować obydwa serwisy za pomocą zwykłej przeglądarki WWW.

5. Serwer za kratkami – chroot jail

Mechanizm chroot(8), bardzo powszechnie stosowany w OpenBSD pomoże nam dodatkowo zabezpieczyć system przed ewentualnymi błędami w kodzie, które mogłyby doprowadzić do kompromitacji całego systemu. Całe środowisko Apache2 zostanie zamknięte w drzewie katalogów `/chroot/httpd`. Podzielę to zadanie na 4 etapy.

1. Należy utworzyć bazę drzewa katalogów oraz zapewnić komunikację z syslogd(8) z wewnątrz środowiska *chroot*:

```
# mkdir -p /chroot/httpd/dev
# echo 'syslogd_flags="-a /chroot/httpd/dev/log"' >> /etc/rc.conf
# kill `cat /var/run/syslog.pid` && syslogd -a /chroot/httpd/dev/log
```

2. Poniżej przedstawiam autorski sposób skopiowania potrzebnych dla działania Apache2 plików do środowiska *chroot*:

```
# cd /tmp
# ktrace -i /usr/local/apache2/http
# kdump |grep NAMI|awk '{print $4}'|sort|uniq|sed 's/"//g' | cpio -p -dLv /chroot/httpd
```

Polecenia *ktrace(1)/kdump(1)* pokazują historię odwołań systemowych, która jest przetwarzana na listę plików kopiowanych do katalogu docelowego.

3. Pozostaje jedynie usunąć zbędne wpisy z bazy użytkowników:

```
# cd /chroot/httpd/etc
# cp /etc/master.passwd . # edycja pliku, pozostawiamy TYLKO wpisy apache i nobody
# pwd_mkdb -d . master.passwd
```

4. Nie zapomnijmy o skopiowaniu zawartości serwisów WWW:

```
# cp -r /usr/local/www /chroot/httpd/usr/local/www
```

Tak przygotowaną instalacje uruchamiamy poleceniem:

```
# /usr/sbin/chroot /chroot/httpd/ /usr/local/apache2/bin/httpd
```

Pozostaje już tylko zadbać o automatyczne startowanie usługi podczas uruchamiania systemu. W tym celu dopisujemy do skryptu `/etc/rc.local` linijki:

```
if [ -d /chroot/httpd -a -x /chroot/httpd/usr/local/apache2/bin/httpd ]; then
        echo -n ' apache2'
        /usr/sbin/chroot /chroot/httpd /usr/local/apache2/bin/httpd
fi
```

6. Sprawdzenie działania

Podstawowe test, to weryfikacja działania serwera po restarcie systemu za pomocą dowolnej przeglądarki WWW. Poniżej znajduje się dokładny zapis sesji telnet pobierającej główną strone serwisu obsługiwanego przez Apache2.

```
ptroja@mamroot:~$ telnet 192.168.1.9 www
Trying 192.168.1.9...
Connected to 192.168.1.9.
Escape character is '^]'.
GET http://www.bss.lab/ 1.1

HTTP/1.1 200 OK
Date: Thu, 20 Jan 2005 10:53:36 GMT
Server: Apache
Last-Modified: Sun, 09 Jan 2005 19:58:26 GMT
ETag: "be7-61-dff37c80"
Accept-Ranges: bytes
Content-Length: 97
Connection: close
Content-Type: text/html

<html><head><title>www.bss.lab testing site</title></head><body>Apache2 4for4
BSS!</body></html>
Connection closed by foreign host.
```

Do przetestowania zabezpieczeń serwera użyłem programu *Nikto*. Program ten testuje odporność serwera WWW poprzez wysyłanie zapytań zgromadzonych w swojej wewnętrznej bazie odwołań, które potencjalnie mogą stanowić lukę do wykorzystania przez włamywaczy. Poniżej znajduje się zapis uruchomienia programu podczas którego (dane na podstawie logów serwera, plik `access.log`) zostało przetestowanych 16 tys. zapytań (czas wykonania 15 minut).

```
ptroja@mamroot:~$ nikto -g -C all -h 192.168.1.9
---------------------------------------------------------------------------
- Nikto 1.32/1.23     -     www.cirt.net
+ Target IP:       192.168.1.9
+ Target Hostname: www.bss.lab
+ Target Port:     80
+ Start Time:      Thu Jan 20 11:43:55 2005
---------------------------------------------------------------------------
+ Server: Apache
+ Allowed HTTP Methods: GET,HEAD,POST,OPTIONS,TRACE
+ HTTP method 'TRACE' is typically only used for debugging. It should be disable
+ /\"><img%20src=\"javascript:alert(document.domain)\"> - The IBM Web Traffic Ex
+ /?Open - This displays a list of all databases on the server. ÊDisable this ca
+ /xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
+ 15943 items checked - 3 item(s) found on remote host(s)
+ End Time:        Thu Jan 20 11:57:53 2005 (838 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

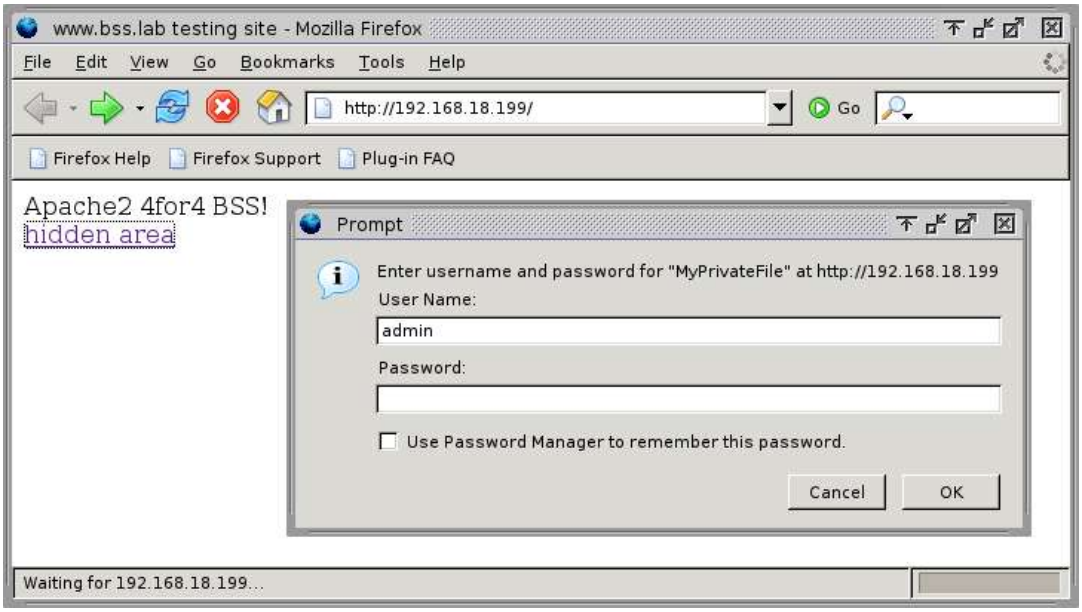
Metoda TRACE, pomimo wprowadzenia zalecanych zabezpieczeń i tak pozostaje dla widoczna dla programu testowego, ale analiza sesji “telnet” pokazuje, że jest to tylko fałszywy alarm:

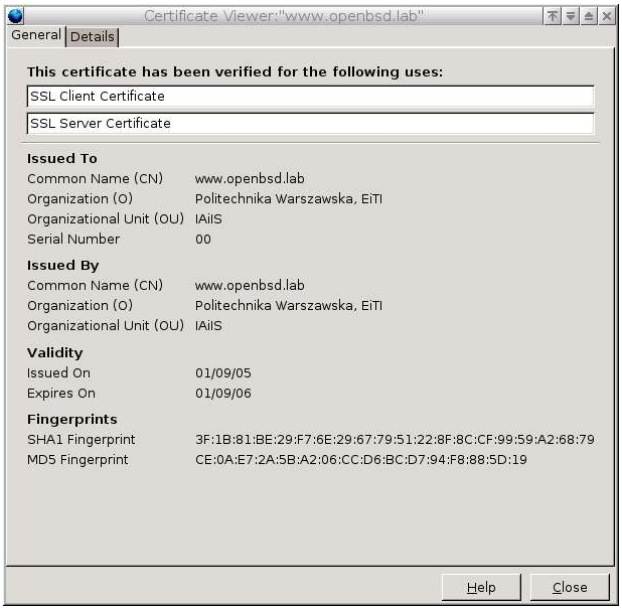
```
ptroja@mamroot:~$ telnet www.bss.lab www
Trying 192.168.1.9...
Connected to www.bss.lab.
Escape character is '^]'.
TRACE /index.html 1.1

HTTP/1.1 403 Forbidden
Date: Thu, 20 Jan 2005 11:21:31 GMT
Server: Apache
Content-Length: 212
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!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 /index.html
on this server.</p>
</body></html>
Connection closed by foreign host.
```

Ze względu na brak obsługi protokołu IPv6 w środowisku testowym, sprawdzenie jego obsługi ograniczyłem tylko do weryfikacji otwartych portów oraz sesji telnet z maszyny, na której pracuje Apache2:

```
# netstat -a -f inet6 -n
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp6       0      0  *.443                  *.*                    LISTEN
tcp6       0      0  *.80                   *.*                    LISTEN

# telnet ::1 80
Trying ::1...
Connected to ::1.
Escape character is '^]'.
GET / 1.1

HTTP/1.1 200 OK
Date: Thu, 20 Jan 2005 11:27:38 GMT
Server: Apache
Last-Modified: Sun, 09 Jan 2005 19:58:26 GMT
ETag: "be7-61-dff37c80"
Accept-Ranges: bytes
Content-Length: 97
Connection: close
Content-Type: text/html

<html><head><title>www.bss.lab testing site</title></head><body>Apache2 4for4
BSS!</body></html>
Connection closed by foreign host.
```

Poniżej zamieszczam dodatkowo zrzuty ekranu z procesu weryfikacji certyfikatu oraz autoryzacji dostępu do chronionego obszaru domeny *www.openbsd.lab*.





7. Podsumowanie

Zaprezentowany sposób instalacji oraz konfiguracji serwera z pewnością nie wyczerpuje kwestii bezpiecznej instalacji serwera WWW w systemie OpenBSD, chociaż przedstawione kroki z pewnością zapewniają instalację na o stopniu bezpieczeństwa. Niestety odbywa się to kosztem funkcjonalności – konfiguracja dynamicznych serwisów WWW wykorzystujących PHP/PERL, korzystających z baz danych SQL wykracza poza ramy tak krótkiego opracowania. Nie została poruszona kwestia zarządzania, analizy oraz archiwizowania logów z serwera – element nie do pominięcia przy wdrażaniu dowolnej bezpiecznej usługi.

Serwer Apache2, pomimo braku akceptacji środowiska OpenBSD może stanowić bardzo kuszące połączenie funkcjonalności i bezpieczeństwa jakie oferuje sam system operacyjny. Jego integracja z mechanizmem zabezpieczeń *chroot(8)* została znacznie uproszczona, dzięki zastosowaniu oryginalnego sposobu kopiowania niezbędnych plików.

8. Bibliografia

Spis linków z wewnątrz dokumentu.

1. Setting Up a Secure Apache 2 Server, http://www.samspublishing.com/articles/article.asp?p=30115
2. Securing Apache 2: Step-by-Step, http://www.securityfocus.com/infocus/1786
3. Setting up a Secure HTTP server with SSL(8), http://www.openbsd.org/faq/faq10.html#HTTPS
4. Locking Down Apache, http://www.bastille-linux.org/jay/Talks/slides-defcon-securing-apache.pdf
5. Netcraft, http://news.netcraft.com/archives/web_server_survey.html

6. Stronghold, http://stronghold.redhat.com/
7. Strony domowe systemów operacyjnych: www.openbsd.org, www.freebsd.org, www.debian.org.
8. Apache Webserver, http://httpd.apache.org/

9. Załącznik A, plik konfiguracji serwera Apache2

```
# ================================================
# Basic settings
# ================================================
# use HTTP+HTTPS over IP+IPv6
Listen 0.0.0.0:80
Listen [::]:80
Listen 0.0.0.0:443
Listen [::]:443
User apache
Group apache
ServerAdmin P.Trojanek@elka.pw.edu.pl
ServerName www.bss.lab
UseCanonicalName Off
ServerSignature Off
HostnameLookups Off
ServerTokens Prod
ServerRoot "/usr/local/apache2"
DocumentRoot "/usr/local/www"
PidFile /usr/local/apache2/logs/httpd.pid
ScoreBoardFile /usr/local/apache2/logs/httpd.scoreboard
<IfModule mod_dir.c>
    DirectoryIndex index.html
</IfModule>

# ================================================
# HTTP and performance settings
# ================================================
Timeout 300
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 15
<IfModule prefork.c>
    MinSpareServers 5
    MaxSpareServers 10
    StartServers 5
    MaxClients 150
    MaxRequestsPerChild 0
</IfModule>

<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{REQUEST_METHOD} ^TRACE
    RewriteRule .* - [F]
</IfModule>

# ================================================
# Access control
# ================================================
<Directory />
Options None
    AllowOverride None
    Order deny,allow
    Deny from all
</Directory>
<Directory "/usr/local/www/www.bss.lab">
    Order allow,deny
    Allow from all
</Directory>
<Directory "/usr/local/www/www.openbsd.lab">
```

```
    Order allow,deny
    Allow from all
</Directory>

AccessFileName .htaccess
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>

# ================================================
# MIME encoding
# ================================================
<IfModule mod_mime.c>
    TypesConfig /usr/local/apache2/conf/mime.types
</IfModule>
DefaultType text/plain
<IfModule mod_mime.c>
    AddEncoding x-compress .Z
    AddEncoding x-gzip .gz .tgz
    AddType application/x-compress .Z
    AddType application/x-gzip .gz .tgz
    AddType application/x-tar .tgz
</IfModule>

# ================================================
# Logs
# ================================================
LogLevel warn
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
ErrorLog /usr/local/apache2/logs/error_log
CustomLog /usr/local/apache2/logs/access_log combined

# ================================================
# Virtual hosts
# ================================================
#NameVirtualHost *
<VirtualHost *:80>
    DocumentRoot "/usr/local/www/www.bss.lab"
    ServerName "www.bss.lab"
    ErrorLog logs/www.bss.lab/error_log
    CustomLog logs/www.bss.lab/access_log combined
    <IfModule mod_rewrite.c>
        RewriteEngine On
        RewriteCond %{REQUEST_METHOD} ^TRACE
        RewriteRule .* - [F]
    </IfModule>
</VirtualHost>

<VirtualHost *:443>
    DocumentRoot "/usr/local/www/www.openbsd.lab"
    ServerName "www.openbsd.lab"
    ErrorLog logs/www.openbsd.lab/error_log
    CustomLog logs/www.openbsd.lab/access_log combined
    SSLEngine on
    SSLCertificateFile    /etc/ssl/www.openbsd.lab.crt
    SSLCertificateKeyFile /etc/ssl/private/www.openbsd.lab.key
    <IfModule mod_rewrite.c>
        RewriteEngine On
        RewriteCond %{REQUEST_METHOD} ^TRACE
        RewriteRule .* - [F]
    </IfModule>
</VirtualHost>
```