Bezpieczna komunikacja

OpenSSL i Stunnel

Bezpieczna komunikacja pomiędzy systemami komputerowymi staje się warunkiem koniecznym do wymiany informacji we współczesnym świecie.

CEZARY GAJDZIŃSKI

www.photocase.de

Istnieje wiele technologii pozwalających na bezpieczną komunikację, np. IPSec, PPTP, SSH, SSL. Każda z nich ma swoje wady i zalety w przypadku tworzenia VPN (Virtual Private Network). Chciałbym przybliżyć technologię SSL i wskazać, jak można ją wykorzystać do tworzenia kanałów komunikacji z szyfrowaniem transmisji i autoryzacją stron.

Załóżmy, że X jest maszyną, której zadaniem jest świadczenie usług dostępu do poczty elektronicznej i konta użytkownika. Wymagamy, aby komunikacja pomiędzy klientem Y i maszyną X (którą nazwiemy teraz serwerem) odbywała się z wykorzystaniem szyfrowanej komunikacji i autoryzacji dostępu. Pominiemy definicję formalną i szczegóły „filozoficzne" bezpiecznej komunikacji, koncentrując się na rozwiązaniu praktycznym.

Zakładamy, że chcemy zabezpieczyć komunikację do serwera X oferującego usługi smtp, imap, pop (i inne). Jednym z możliwych rozwiązań jest zastosowanie bezpiecznych serwisów smtpd + TLS, imaps, pops, czy ssh dla użytkownika. Wadą tego jest jednak różnorodność rozwiązań. Każdy z tych „bezpiecznych" serwerów ma odrębne wymagania i konfigurację.

SSL pozwala na zastosowanie jednego mechanizmu dla popularnych serwisów. W celu zabezpieczenia komunikacji zajmiemy się techniką stunnel. Pomysł jest prosty – zamiast zmieniać usługi, pozwólmy ich klientom decydować, czy komunikacja ma być bezpieczna. SSL i stunnel pozwalają na jednoczesną autoryzację i szyfrowanie komunikacji. Istotne jest zrozumienie podstawowego faktu: to klient decyduje o bezpieczeństwie. Dlatego też musi być spełniony jeden warunek – oprogramowanie klienta powinno posiadać „wbudowaną" obsługę SSL.

Oto warunki niezbędne dla bezpiecznej komunikacji:

1. Klient „rozumie" i obsługuje technologię SSL.
2. Serwer ma uruchomioną usługę stunnel i pozostałe usługi dla klientów.
3. Istnieje centrum autoryzacyjne (CA).

Dla spełnienia trzeciego warunku najprostszym rozwiązaniem jest wygenerowanie certyfikatów typu self-signed, czyli takich, które podpisujemy sami. Nie jest to najbezpieczniejszy pomysł, ale całkowicie wystarczający do testów. Jednak dopiero stworzenie normalnego, pełnego CA daje pełną kontrolę nad wystawionymi certyfikatami i możliwość ich odwołania.

Załóżmy więc, że mamy gotowy pewien zbiór klientów spełniających pierwszy warunek (np. dla pop, imap, smtp). Od strony serwera udostępniamy usługi smtp, pop, imap i stunnel (np. poprzez xinetd). Istotne jest, aby te usługi (oprócz stunnel) nasłuchiwały wyłącznie na adresie 127.0.0.1 (loopback). Usługa xinetd pozwala na szybką konfigurację, uwzględniającą powyższe założenie.

Stunnel, jak już wspomnieliśmy, wymaga do poprawnego działania utworzenia centrum autoryzacyjnego, wygenerowania dla usługi stunnel certyfikatu i przygotowania poprawnego pliku konfiguracyjnego zawierającego definicje dla poszczególnych usług (pop, imap, smtp).

Załóżmy tutaj, że korzystamy z OpenSSL, który został zainstalowany w katalogu */usr/local/ssl*. Poniższe punkty opisują w szczegółach konfigurację, która zapewni opisaną funkcjonalność.

Wygenerowanie certyfikatu i klucza dla CA

Przechodzimy (jako użytkownik root) do katalogu */usr/local/ssl/misc/* i edytujemy plik *CA.pl* (lub *CA.sh*) oraz zmieniamy *DAYS='-days 365'* na *DAYS='-days 1825'*. Zmiana ta wydłuża ważność certyfikatów. Teraz generujemy podstawowe pliki centrum autoryzacyjnego CA:

```
./CA.sh -newca
```

Należy pamiętać, aby poprawnie odpowiedzieć na pytania, które opisywać będą nasze CA. W katalogu */usr/local/ssl/misc/demoCA/* powstanie plik:

```
cacert.pem – certyfikat CA
```

a w jego podkatalogu *private/:*

```
cakey.pem – klucz prywatny CA.
```

Stworzyliśmy w ten sposób pełnoprawne centrum autoryzacyjne!

Instalacja stunnel

Pobieramy ze strony *http://www.stunnel.org* odpowiednią wersję kodu źródłowego, rozpakowujemy w wybranym katalogu i instalujemy:

```
./configure --sysconfdir=/etc
make
make install
```

Parametr */etc* dla *--sysconfdir* jest dowolny.

Konfigurujemy stunnel

Tam, gdzie rozpakowany został stunnel, w podkatalogu tools/ znajduje się też plik *stunnel.cnf*. Kopiujemy go do */usr/local/ssl/certs/*. Na jego podstawie będzie wygenerowany certyfikat dla usługi stunnel. Przechodzimy teraz do */usr/local/ssl/certs/* i piszemy:

```
openssl req -new -nodes -out ➥
req.pem -keyout key.pem -config ➥
stunnel.cnf
```

Wpisujemy właściwe dane, odpowiadając na pytania. Najważniejsze jest to, żeby w miejscu gdzie pytani jesteśmy o FQDN (Common Name), wpisać FQDN serwera. Utworzone w ten sposób zostały pliki key.pem (klucz prywatny) i req.pem (prośba o wystawienie certyfikatu).

Podpisanie pliku req.pem

Kopiujemy req.pem do /usr/local/ssl/misc/, ale zapisujemy go jako newreq.pem (parametr dla CA.pl). Teraz wchodzimy do /usr/local/ssl/misc/ i wydajemy polecenie:

```
./CA.pl -sign
```

OpenSSL podpisze plik *newreq.pem* i zapyta o hasło klucza prywatnego CA. W rezultacie zostanie wygenerowany nowy plik – *newcert.pem*. Jest to podpisany przez nasze CA certyfikat dla usługi stunnel. Przenosimy newcert.pem z powrotem do */usr/local/ssl/certs/*, tworzymy pusty plik o nazwie stunnel.pem (touch stunnel.pem) z prawami *0600*. Sprawdzamy katalog roboczy – powinniśmy być w katalogu */usr/local/ssl/certs*.

Teraz dokonujemy konkatenacji (stunnel musi mieć klucz i certyfikat w jednym pliku, w tej kolejności):

```
cat key.pem > stunnel.pem
echo '' > stunnel.pem
cat newcert.pem > stunnel.pem
```

Zostawiamy tylko plik stunnel.pem, pozostałe z nich kasujemy.

Składamy wszystko razem

W /etc/services powinny znajdować się następujące wpisy:

```
pop3s 995/tcp
imaps 993/tcp
ssmtp 465/tcp
```

Najpierw plik stunnel.pem przenosimy do /etc/stunnel/, a następnie tworzymy katalog /var/run/stunnel/ i zmieniamy jego właściciela na użytkownika nobody i grupy nobody.

```
mkdir /var/run/stunnel
chown nobody.nobody ➥
/var/run/stunnel/
```

Na końcu kopiujemy certyfikat CA do /etc/stunnel:

```
cp /usr/local/ssl/misc/demoCA➥
/cacert.pem /etc/stunnel/
```

Teraz tworzymy plik stunnel.conf; na początek przechodzimy do /etc/stunnel/ i tworzymy pusty plik stunnel.conf:

```
cd /etc/stunnel/
touch stunnel.conf
```

Umieszczamy w nim następujące wpisy:

```
CAfile = /etc/stunnel/cacert.pem ➥
#certyfikat centrum ➥
autoryzacyjnego CA
cert = /etc/stunnel/stunnel.pem ➥
#certyfikat usługi stunnel
debug = 7          #poziom logowania
output = /var/log/stunnel.log ➥
#plik logowania
chroot = /var/run/stunnel/ ➥
#katalog roboczy usługi stunnel
pid = /stunnel.pid ➥
#ścieżka względna pliku .pid ➥
(względem chroot)
setuid = nobody
setgid = nobody
[pop3s]
accept = 995
connect = 110

[imaps]
accept = 993
connect = 143

[ssmtp]
accept = 465
connect = 25
```

Wpisujemy polecenie *stunnel*. W pliku /var/log/stunnel.log znajdują się potrzebne informacje diagnostyczne. Jeśli wszystko wydaje się poprawne, to polecenie *netstat -a* powinno pokazać odpowiednie porty (995/993/465) jako oczekujące na połączenie. Możliwe jest również uruchomienie usługi stunnel ręcznie, na przykład:

```
stunnel -d 192.168.0.1:993 -r ➥
127.0.0.1:imap2
```

Pozwala to na diagnostykę poszczególnych serwisów. Pamiętajmy również, że klucze i certyfikaty powinny mieć odpowiednie prawa dostępu (0600 dla użytkownika root).

Wnioski

Powodzenie technologii opartej na stunnel zależy od liczby dostępnych aplikacji klienckich dla poszczególnych usług. Dlatego programiści powinni w swoich aplikacjach zapewniać obsługę SSL. Nie jest to zbyt trudne, ponieważ OpenSSL posiada własne API umożliwiające łatwe oprogramowanie obsługi protokołu SSL.

Własne CA pozwala na wydawanie certyfikatów zainteresowanym osobom. Należy pamiętać o pewnych istotnych szczegółach związanych z wyborem formatu certyfikatów w zależności od aplikacji i usługi (np. X.509, PKCS#7, PKCS#12). Stunnel świetnie sprawdza się w środowiskach heterogenicznych, pewną trudnością może być jedynie konfiguracja oprogramowania klientów. ■

AUTOR

Cezary Gajdziński od wielu lat zajmuje się problemami związanymi z bezpieczeństwem systemów linuksowych i uniksowych oraz bezpieczną wymianą informacji. Obecnie pracuje w Systemics Poland, jest dostępny pod adresem e-mail: cezary.gajdzinski@systemics.pl.