Bezpieczna poczta elektroniczna – tunelowanie po protokole SSL

Paweł Krawczyk

2 lipca 2001

Spis treści

1 Wstęp

Poczta elektroniczna jest jedną z najczęściej wykorzystywanych usług internetowych, korzysta z niej praktycznie każdy. Jest to także usługa bardzo osobista — każdy ma własny adres email, a czytanie cudzej poczty jest jednym z największych przestępstw przeciwko netykiecie. Poufność korespondencji, którą chronić można za pomocą PGP, to jednak tylko jedna strona medalu. Drugim największym problemem związanym z pocztą elektroniczną jest bezpieczeństwo protokołu POP3 używanego do jej ściągania z serwera.

POP3 wymaga zalogowania do serwera, przy czym hasło oraz login użytkownika są przesyłane otwartym tekstem i łatwe do podsłuchania. Do niedawna jednym z głównych źródeł takich informacji uzyskiwanych za pomocą snifferów był *telnet*, jednak w ciągu ostatnich kilku lat został on praktycznie wyeliminowany przez bezpieczne *SSH*. Problem jednak nadal istnieje, jako że ogromna większość ludzi nadal ściąga swoją pocztę po POP3 bez żadnej dodatkowej ochrony.

W międzyczasie wynaleziono rozszerzenia POP3 (takie jak APOP), mające za zadanie zabezpieczenie procesu logowania przed podsłuchaniem, ale mają one szereg wad wychodzących w praktyce. Jedną z nich jest także to, że nie są one zbyt chętnie wdrażane przez producentów oprogramowani klienckiego, co niestety jest dość poważnym argumentem w dobie dominacji Windows na stacjach roboczych.

W chwili obecnej najlepszym rozwiązaniem, posiadającym szereg zalet i, co istotne, wspieranym przez przemysł komputerowy jest protokół SSL. Z punktu widzenia klienta jest to przeważnie kwestia jednego przełącznika w konfiguracji programu pocztowego. Postawienie serwera POP3/SSL wymaga trochę więcej zaangażowania od administratora, ale wysiłek ten z pewnością się opłaci.

Główne zalety tunelowania POP3/SSL to:

- szyfrowana jest całość transmisji, włącznie z nagłówkami i treścią ściąganych listów
- protokół SSL jest powszechnie uznanym standardem, dostępnym na praktycznie wszystkie używane obecnie platformy
- z kryptograficznego punktu widzenia SSL jest protokołem sprawdzonym i bezpiecznym

2 Oprogramowanie

Dla potrzeb tego artykułu nie ma większego znaczenia, z jakiego systemu operacyjnego korzystamy, pod warunkiem że jest to system unixowy ;). Wykorzystywane oprogramowanie jest dostępne w postaci źródeł i kompiluje się na wszystkich popularnych odmianach Unixa, z Linuxem i BSD na czele.

Oprzemy się na następujących pakietach oprogramowania:

- **OpenSSL**
  Biblioteki oraz aplikacje kryptograficzne, stanowiące serce protokołu SSL. Dostępne w postaci binarnej dla wszystkich dystrybucji Linuxa oraz BSD (pakiet *openssl*), a w przypadku OpenBSD jest dostępny od razu w standardowej dystrybucji. W przypadku innych systemów operacyjnych konieczne może być skompilowanie pakietu ze źródeł dostępnych na wielu serwerach (lista na końcu artykułu).

- **Stunnel**
  Uniwersalny demon *proxy*, pozwalający na tunelowanie po SSL praktycznie dowolnych protokołów, z POP3 włącznie. W największym skrócie, *stunnel* potrafi przyjąć połączenie SSL i przesyłać rozszyfrowane dane dalej, do zadanego serwera, a następnie zwracać otrzymane wyniki z powrotem przez bezpieczny tunel. Również dostępny w postaci gotowych pakietów oraz jako kod źródłowy.

- **Solid–POP3**
  Tunelowanie POP3 nie wymaga żadnej „świadomości" tego faktu ze strony demona serwującego ten protokół, ale jeśli nie wybrałeś jeszcze żadnego innego, to polecam właśnie *spop3d*.

Wszystkie wymienione wyżej programy są dostępne jako *open–source*, są rozwijane od dłuższego czasu i nie miały w przeszłości problemów z bezpieczeństwem.

3 Garść teorii

3.1 Tunelowanie POP3/SSL

Tunelowanie POP3 po SSL odbywa się w sposób przezroczysty dla tego pierwszego protokołu. Nie wie on nic o tym, że faktycznie cała konwersacja odbywa się wewnątrz szyfrowanego tunelu SSL. Jest to bardzo wygodne, ponieważ nie wymaga żadnego szczególnego wsparcia ze strony samego serwera POP3 i pozwala na łatwe adaptowanie tego rozwiązania dla innych protokołów pocztowych, np. SMTP lub IMAP.

Konfiguracja nie wymaga więc żadnych zmian w serwerze POP3, działającym na porcie 110. Zamiast tego na porcie 995 (zarezerwowanym dla POP3/SSL) uruchamiamy *stunnel*, przyjmujący zaszyfrowane połączenia od klientów. Połączenie to jest rozszyfrowywane przez *stunnela* i przekierowywane, już jako zwykły protokół POP3, na port 110 tego samego serwera.

Chroniony w ten sposób jest najbardziej narażony na podsłuch odcinek pomiędzy klientem a serwerem, zakładamy natomiast oczywiście, że połączenia w ramach samego serwera są bezpieczne. Odpowiedzi serwera wracają ponownie do *stunnela*, który wysyła je do klienta w tym samym zaszyfrowanym tunelu SSL. Klient nie wie nic o tych manipulacjach — dla niego po prostu wygląda to jak protokół POP3 tunelowany po SSL na porcie 995.

3.2 Certyfikaty SSL

Protokół SSL, pomimo że stosunkowo prosty, wykorzystuje do uwierzytelnienia złożony standard jakim jest X.509, będący częścią Infrastruktury Klucza Publicznego (*Public Key Infrastructure*). Jego celem jest upewnienie klienta, że faktycznie łączy się z tym serwerem, z którym chciał się połączyć. Certyfikaty X.509 opierają się o kryptografię klucza publicznego i są hierarchiczne, tzn. organizacje wyższego poziomu mogą poświadczać certyfikaty organizacji niższego poziomu. Weryfikacja legalności certyfikatu otrzymanego na samym dole takiego drzewa certyfikacji wymaga sprawdzenia całej ścieżki, aż do samej góry.

Koncepcje stojące za X.509 są stosunkowo proste, lecz złożoność samego standardu bardzo utrudnia ich zauważenie i, paradoksalnie, ten mały krok stanowi z reguły najtrudniejszą część konfiguracji serwera SSL. W 90% bowiem procentach przypadków cała ogromna infrastruktura X.509 nie jest nam potrzebna, wykorzystujemy bowiem drobny jej ułamek.

Wystarczy tylko wiedzieć, że będziemy musieli wygenerować na własne potrzeby dwa certyfikaty. Licząc od końca, będzie to certyfikat (klucz publiczny) samego serwera POP3/SSL oraz drugi, certyfikat naszego mikrourzędu certyfikującego (*Certifying Authority*), którym poświadczymy certyfikat serwera.

4 Konfiguracja SSL

Zaczniemy od przygotowania certyfikatów dla serwera POP3/SSL. Warto pamiętać, że certyfikat X.509 jest w tym wypadku związany z nazwą serwera i łączący się z nami klient będzie automatycznie porównywał wskazaną mu nazwę maszyny z nazwą przedstawioną mu w certyfikacie. Dlatego ważne jest, by nazwa skonfigurowana w certyfikacie była taka sama, jak ta podawana klientom, np. *mail.domena.pl*. Z drugiej strony, dzięki temu możemy ten sam

certyfikat wykorzystać do wielu różnych usług na tej samej maszynie (np. SMTP/SSL, IMAP/SSL).

4.1 Certyfikat CA

Certyfikat jest tak na prawdę kluczem publicznym serwera, wraz z jego opisem oraz sygnaturą instytucji wyższego poziomu. Występuje zawsze w parze z kluczem prywatnym, od którego wygenerowania zaczniemy. Stworzymy klucz RSA o długości 1024 bitów i zapiszemy go do pliku *cakey.pem*:

```
$ openssl genrsa -out cakey.pem 1024
Generating RSA private key, 1024 bit long modulus
.................++++++
.....................++++++
e is 65537 (0x10001)
```

Następnie, z klucza prywatnego generujemy klucz publiczny, dokładając do niego informacje o serwerze. To wszystko stanowi *Certificate Signing Request*, czyli po prostu niepodpisany jeszcze certyfikat X.509.

```
$ openssl req -new -key cakey.pem -out cacsr.pem
Using configuration from /usr/lib/ssl/openssl.cnf
-----
Country Name (2 letter code) [AU]:PL
State or Province Name (full name) [Some-State]:Malopolska
Locality Name (eg, city) []:Krakow
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ABA sp. z o.o.
Organizational Unit Name (eg, section) []:.
Common Name (eg, YOUR name) []:ABA sp z o.o
Email Address []:hostmaster@aba.krakow.pl
```

Jak widać, podajemy podstawowe informacje o samym serwerze oraz jego właścicielu. W „prawdziwym” X.509 organizacja występująca o oficjalnie podpisany certyfikat musi dowieść, że dane te są prawdziwe i że ma prawo posługiwać się podaną nazwą. Właśnie formalne potwierdzenie prawdziwości tych danych jest sednem procesu certyfikacji.

W tej chwili powinniśmy zgłosić nasz CSR do podpisania, jest to jednak proces kosztowny i niepotrzebny, jeśli stawiamy serwer na potrzeby firmy, a zależy nam głównie na poufności transmisji. Standard SSL przewiduje na szczęście coś takiego, jak certyfikaty *self–signed*, czyli podpisane przez tę samą osobę, która wystawia CSR. W tym celu bierzemy podpisywany CSR, podpisujący klucz prywatny i wynik zapisujemy do pliku *cacert.pem*.

```
$ openssl x509 -req -in cacsr.pem -signkey cakey.pem -out cacert.pem
```

```
Signature ok
subject=/C=PL/ST=Malopolska/L=Krakow/O=ABA sp. z o.o.
        /CN=mail.aba.krakow.pl/Email=hostmaster@aba.krakow.pl
Getting Private key
```

Tego certyfikatu CA możemy teraz używać wielokrotnie do podpisywania różnych certyfikatów roboczych dla poszczególnych usług. W tym wypadku zrobimy to dla serwera *mail.aba.krakow.pl*, dla którego musimy powtórzyć cały proces z pewnymi różnicami.

4.2 Certyfikat serwera

Tak jak poprzednio, generujemy klucz prywany RSA i zapisujemy go do pliku *pop3key.pem*:

```
$ openssl genrsa -out pop3key.pem 1024
```

Następnie generujemy CSR podając te same dane firmy, z jedną różnicą. Zamiast nazwy firmy w polu *Common name* podajemy nazwę serwera:

```
$ openssl req -new -key pop3key.pem -out pop3csr.pem
...
Common Name (eg, YOUR name) []:mail.aba.krakow.pl
...
```

Pole to będzie później wykorzystywane przez klientów do porównania z nazwą serwera, z którym się połączyli.

Następny krok to już tylko złożenie podpisu. Nasz mikrourząd certyfikujący podpisze swoim kluczem nowy certyfikat serwera *mail.aba.krakow.pl*:

```
$ openssl x509 -CA cacert.pem -CAkey cakey.pem -req -CAcreateserial \
               -in pop3csr.pem -out pop3cert.pem
Signature ok
subject=/C=PL/ST=Malopolska/L=Krakow/O=ABA sp z o.o.
        /CN=mail.aba.krakow.pl/Email=hostmaster@aba.krakow.pl
Getting CA Private Key
```

Do złożenia podpisu wykorzystujemy jak widać klucz prywatny CA (to on faktycznie składa podpis), klucz publiczny CA, czyli jego certyfikat (są tam dane CA). Dane wejściowe to CSR serwera, a na wyjściu otrzymujemy jego podpisany certyfikat. Opcja `-CAcreateserial` powoduje stworzenie pliku zawierającego numery kolejnych certyfikatów, wystawianych przez tego CA. Ta szczypta biurokracji jest tutaj niezbędna.

5 Konfiguracja Stunnela

Teraz musimy przygotować środowisko do uruchomienia Stunnela. W tym celu połączymy dwa pliki, klucz oraz certyfikat serwera w jeden, umieszczony przykładowo w katalogu */var/pki*.

```
$ mkdir /var/pki
$ cat pop3key.pem pop3cert.pem >>/var/pki/pop3.pem
```

Następnie założymy grupę i użytkownika *stunnel*, aby demon po zajęciu odpowiednich portów porzucił uprawnienia *roota*, zmniejszając w ten sposób ryzyko naruszenia bezpieczeństwa serwera:

```
$ groupadd stunnel
$ useradd stunnel -g stunnel -d /dev/null -s /bin/false
$ chown stunnel.stunnel /var/pki/pop3.pem
$ chmod 400 /var/pki/pop3.pem
```

Teraz wystarczy już tylko na uruchomić *stunnel*, wskazując mu, na jakim porcie ma przyjmować połączenia SSL (995) oraz na jaki port ma kierować rozszyfrowany protokół POP3 (110). Dokładny opis tych opcji można znaleźć w manualu *stunnel(8)*, więc ograniczymy się do przedstawienia gotowej linii poleceń:

```
$ stunnel -g stunnel -s stunnel -N stunnel -d 995 -p /var/pki/pop3.pem -r 110
```

Program zgłasza komunikaty na temat swojego działania za pomocą *sysloga*. W razie problemów można go uruchomić dodatkowo z opcjami `-f` i `-D`, które zwiększą ilośc informacji zgłaszanych przez program.

Od tej pory klienci powinni móc się już łączyć na ten port i ściągać przez niego pocztę. Najprostszy test to nawiązanie takiego połączenia ręcznie za pomocą klienta SSL:

```
$ openssl s_client -quiet -connect localhost:995
CONNECTED(00000003)
...
+OK Solid POP3 server ready
```

Fakt zgłoszenia się serwera POP3 jest wystarczającym dowodem na poprawność instalacji. Jeśli usługa nadal nie działa poprawnie, to przyczyn należy szukać gdzie indziej — na przykład w konfiguracji filtrów pakietowych lub programów pocztowych.

6 Adresy

- **OpenSSL** http://www.openssl.org/
- **Stunnel** http://www.stunnel.org/
- **Solid–POP3** http://solidpop3d.pld.org.pl/

7 FAQ

7.1 Jak uruchomić *stunnel* z *inetd*?

Należy opuścić opcje `-d`, domyślnie *stunnel* działa właśnie w trybie *inetd*. Jeśli nie chcemy w ogóle udostępniać demona POP3 na porcie 110, możemy skonfigurować *stunnel* tak, by sam uruchamiał potrzebny program przy pomocy opcjie `-l`.

W takim wypadku konfiguracja *inetd* będzie wyglądać następująco:

```
pop3s     stream  tcp     nowait  root    stunnel stunnel
                           -l /usr/sbin/spop3d -p /var/pki/spop3d.pem
```

7.2 Dlaczego demon POP3 zawsze loguje połączenie z *localhost*?

Połączenie do demona POP3 nie przychodzi bezpośrednio od klienta, tak na prawdę inicjuje je *stunnel* i jest to zupełnie niezależne połączenie TCP. Klient pocztowy łączy się natomiast i wymienia informacje z samym *stunnelem* i to w jego logach należy szukać informacji o adresie IP klienta.

Użytkownicy Linuxa mogą ten problem obejść jeszcze w jeden sposób, a mianowice stosując opcję `-T` (*transparent proxy*). Posiada ona jednak pewne ograniczenia, dokładniej opisane w manualu *stunnel(8)*.

7.3 Czy można odciążyć serwer WWW stawiając przed nim Linuxa ze *stunnelem*

?

Podejrzewam że tak, ale nigdy tego nie robiłem. Na pewno ograniczy to mocno możliwości kontroli dostępu zapewniane przez certyfikaty SSL, bo serwer nic nie będzie o nich wiedział.

Z drugiej jednak strony wydaje mi się, że w większości przypadków, kiedy rola serwera SSL ogranicza się do przyjęcia zamówienia z formularza takie

rozwiązanie może być bardzo dobrym pomysłem na odciążenie serwera. Warto dodać, że podobne rozwiązanie komercyjnie oferował Intel (obecnie sprzedaje to Compsq oraz HP).

7.4 Jak wystawić po SSL na routerze POP3 z wewnętrznego serwera?

Na przykład tak:

```
nice -15 /usr/sbin/stunnel -g stunnel -s stunnel -N stunnel -d 995 \
                               -p /var/pki/ssl.pem -r 10.1.1.2:110
```

Nie jest wymagane żadne dodatkowe przekierowanie portów itp., poza oczywiście otwarciem portu 995 na filtrze pakietowym, który oczywiście mamy włączony ;).

8 Od autora

8.1 Uwagi końcowe

Zdaję sobie sprawę, że mogłem popełnić w tym artykule błędy, rzeczowe i inne, w takim wypadku będę wdzięczny za ich wskazanie. Komentarze oraz uzupełnienia są również mile widziane, znajdą się one w kolejnych wersjach tego dokumentu.

8.2 Zmiany

- **Wersja 0.3** kolejne pozycje w FAQ
- **Wersja 0.2** dodane pierwsze FAQ
- **Wersja 0.1** pierwsza wersja tego artykułu

Paweł Krawczyk
kravietz@aba.krakow.pl

ABA sp. z o.o.
ul. Bociana 6
31-231 Kraków