Programowanie sieciowe

mgr inż. Krzysztof Rychlicki-Kicior

Protokoły i standardy

- SMTP/SMTP-AUTH
- FTP/FTPS
- Protokoły a bezpieczeństwo − SSL i TLS
- Secure Shell (SSH)
- WebSockets

SMTP

- Simple Mail Transfer Protocol − stanowy protokół do wymiany poczty elektronicznej
- Choć umożliwia także pobieranie poczty, to w tym celu jest używany praktycznie wyłącznie pomiędzy serwerami pocztowymi − zwyczajowo jest on kojarzony z wysyłaniem wiadomości przez klientów poczty
- Niezabezpieczony SMTP wykorzystuje port 25 − w Polsce, na łączach Orange (Neostrada) port ten jest zablokowany od 2009 roku

SMTP – przykład komunikacji

220 smtp.serwer.com ESMTP Postfix
HELO klient.serwer.com
250 smtp.serwer.com
MAIL FROM: <krzysztof@example.com>
250 Sender <krzysztof@example.com> Ok
RCPT TO: <odbiorca@example.com>
250 Recipient <recipient@example.com> Ok
DATA
354 Ok Send Data ending with <CRLF>.<CRLF>
From: krzysztof@example.com
To: odbiorca@example.com
Subject: Test
Testowa wiadomość
.
server: 250 Message received:
20040120203404.CCCC18555.smtp.serwer.com@klient.serwer.com
QUIT
221 smtp.serwer.com ESMTP server closing connection

SMTP

- Powyższy mechanizm, choć skuteczny i prosty, jest zupełnie niezabezpieczony
- Brakuje weryfikacji nadawcy − niegdyś standardowym zabezpieczeniem była weryfikacja za pomocą ograniczenia lokalizacji, jednak w obecnych czasach ograniczałoby to znacząco możliwości wysyłania poczty
- Oprócz tego transmisja jest zupełnie niezabezpieczona i można ją podsłuchać, wykonać atak man-in-the-middle, itd.

SMTP-AUTH

- Rozszerzenie protokołu SMTP, które pozwala na podanie danych uwierzytelniających

220 smtp.serwer.com ESMTP Server

EHLO klient.serwer.com

250-smtp.serwer.com Hello klient.serwer.com

250 AUTH GSSAPI DIGEST-MD5 PLAIN

250 AUTH PLAIN dGVzdAB0ZXN0ADEyMzQ=

235 2.7.0 Authentication successful

- EHLO zamiast HELO oznacza zastosowanie rozszerzonego SMTP (*Extended SMTP*), co pozwala na użycie uwierzytelnienia

FTP

- File Transfer Protocol – stanowy protokół do transferu plików
- Umożliwia przeglądanie, pobieranie, dodawanie i usuwanie wybranych zasobów serwera dla użytkowników zalogowanych (uwierzytelnionych), jak i anonimowych
- Domyślnie działa na porcie 21

FTP

220 Hello

USER krzysztof

331 Password required to access user account krzysztof

PASS haslo

230 Logged in.

CWD folder

250 "/home/krzysztof/folder" is new working directory.

PORT 10,3,0,100,4,15

200 PORT command successful

LIST

150 Opening ASCII mode data connection for /bin/ls

FTP

226 Listing Completed.

PORT 10,3,0,100,4,16

200 PORT command successful.

RETR plik.txt

150 Opening ASCII mode data connection for plik.txt

226 Transfer completed.

QUIT

221 Goodbye.

- Wysyłanie plików jest możliwe za pomocą polecenia STOR.

FTP

- Połączenie do transferu danych od serwera do klienta nie zawsze jest możliwe (np. z powodu firewalla)

- Można wtedy włączyć tryb pasywny, w którym to klient łączy się z serwerem − w takiej sytuacji serwer musi udostępnić adres IP i port, pod którym można nawiązać połączenie:

PASV

Entering Passive Mode (81,16,62,36,100,111)

FTP

- Podobnie, jak w przypadku SMTP czy SMTP-AUTH, protokół FTP sam z siebie nie odpowiada za szyfrowanie połączenia, co stanowi oczywiście poważny problem bezpieczeństwa

Protokoły a bezpieczeństwo

- SMTP (nawet w wersji SMTP-AUTH) stanowi doskonały przykład dobrze działającego protokołu, który jest jednocześnie kompletnie niezabezpieczony
- Zwłaszcza w dobie sieci bezprzewodowych, po uzyskaniu dostępu nawet do dobrze zabezpieczonej sieci WiFi, można przechwycić całą transmisję, jaka się w niej odbywa
- Brak szyfrowania transmisji na poziomie protokołu warstwy aplikacji może doprowadzić do bardzo negatywnych konsekwencji

Protokoły a bezpieczeństwo

- Istnieją dwie zasadnicze metody zapewniania bezpieczeństwa transmisji:
  - Stosowanie protokołu Transport Layer Security (TLS – dawniej Secure Sockets Layer, SSL)
  - Stosowanie protokołu SSH (Secure Shell)

TLS

- TLS nie jest stosowany do jednej, konkretnej aplikacji/ protokołu – jest uniwersalny i w praktyce jest stosowany w wielu różnych sytuacjach:
  - HTTPS – HTTP over SSL (port 443)
  - FTPS – FTP over SSL (port 990)
  - SMTPS – SMTP over SSL (dawniej port 465, później 587)

TLS a SSL

- Dawniej stosowano jedynie oznaczenie SSL (mimo że ostatnia wersja SSL, 3.0, została wprowadzona w roku 1998) i do niedawna był to de facto obowiązujący standard
- W 2014 roku został odkryty błąd bezpieczeństwa, skutkujący możliwością wykonania ataku o nazwie POODLE, dla którego jedyną możliwością było wyłączenie protokołu SSL
- Od tamtej pory, wszelkie serwery aplikacji wykorzystujące protokół SSL musiały zostać zmodernizowane tak, aby używały TLS w wersji minimum 1.0
- Różnica w nazewnictwie wynika również z braku kompatybilności wstecznej i możliwości downgrade'u połączenia z TLS 1.0 na SSL 3.0

TLS

- TLS pełni trzy główne funkcje:
  - szyfruje połączenie, przez co nawet przechwycenie całej transmisji nie daje możliwości jej odszyfrowania
  - jest w stanie potwierdzić tożsamość obu stron
  - zapewnia integralność danych (ingerencja choćby w najmniejszym stopniu powoduje zakłócenie całej transmisji)

Schemat połączenia TLS

- Krokiem wstępnym dla nawiązania połączenia jest wskazanie przez klienta woli nawiązania bezpiecznego połączenia. Osiąga się to na dwa sposoby:
  - Klient łączy się z określonym portem, który z założenia akceptuje tylko bezpieczne połączenia
  - Po nawiązaniu niezabezpieczonego połączenia, przed przesłaniem jakichkolwiek wrażliwych danych, następuje przesłanie komunikatu o przejściu w tryb zabezpieczony (np. STARTTLS w protokole ESMTP)

TLS – negocjacja

- Gdy zarówno klient, jak i serwer, mają pewność, że ma zostać nawiązane bezpieczne połączenie, dochodzi do etapu negocjacji

1. Klient przesyła listę obsługiwanych metod szyfrowania i funkcji haszujących
2. Serwer wybiera jedną z obsługiwanych metod i funkcji, a następnie przesyła ją do klienta. Serwer przesyła zwykle również certyfikat potwierdzający jego tożsamość, składający się z:
   - nazwy serwera
   - tożsamości zaufanego centrum certyfikacji
   - klucza publicznego przypisanego do serwera

TLS – negocjacja

3. Klient akceptuje (bądź odrzuca) certyfikat serwera

4. Dochodzi do wygenerowania klucza sesji, który posłuży do symetrycznego szyfrowania dalszej komunikacji

5. Najprostszy sposób ustalenia klucza sesji polega na wygenerowaniu przez klienta liczby losowej, zaszyfrowaniu jej kluczem publicznym serwera i przesłaniu jej do niego

6. Na podstawie uzyskanej losowej liczby dochodzi do wygenerowania unikalnego klucza sesji

TLS

- Z uwagi na fakt, że TLS jest powszechnie stosowanym standardem, podlega on licznym analizom i próbom ataku

- W związku z tym, tworząc aplikacje i protokoły oparte o protokół TLS i wymagające odpowiedniego poziomu bezpieczeństwa, konieczne jest zapewnianie TLS minimum w wersji 1.1

Secure Shell

- Secure Shell (SSH) to usługa, która pozwala na logowanie się i kontrolę zdalnych komputerów w sposób bezpieczny
- Jest następcą telnetu
- Działa na porcie 22

SSH

- Sama w sobie, usługa SSH ma szerokie zastosowania, pozwalając na wymianę tekstowych danych z serwerem. Klient SSH stanowi podstawowe narzędzie pracy każdego administratora (przynajmniej z rodziny *nix ☺)

- Ważny aspekt stosowania stanowi tunelowanie

- Tunelowanie pozwala na utworzenie bezpiecznego kanału komunikacji, za pomocą którego można przesyłać dane niezabezpieczone

SSH

- Przykład: chcemy skorzystać z bazy danych, która jest dostępna tylko lokalnie na serwerze (nie jest uruchomiona na interfejsie publicznym)

- Rozwiązanie: tworzymy tunel SSH, który połączy nas z bazą danych:

```
ssh -L 10001:localhost:5432 krzysztof@serwer.com
```

- Następnie, z komputera lokalnego, wystarczy nawiązać połączenie z określonym portem (10001), a wszelka komunikacja trafi, za pośrednictwem tunelu SSH, do serwera bazy danych

SSH

- Był to przykład lokalnego przekierowania portów
- W niektórych sytuacjach konieczne jest zdalne przekierowanie portów
- Problem: chcę, aby mój kolega z firmy był w stanie obejrzeć moją aplikację, którą mam uruchomioną teraz, lokalnie, na porcie 8080
- Na zajęciach mamy dostęp do sieci uczelnianej i nie ma możliwości udostępnienia mojego komputera (jak i żadnego serwera) na zewnątrz

SSH

- Rozwiązanie wymaga dostępu SSH do serwera, który jest publicznie dostępny

- Wystarczy uruchomić polecenie:

```
ssh -R 10002:localhost:8080 krzysztof@serwer.com
```

- Gdy kolega wejdzie na adres http://serwer.com:10002/, zobaczy aplikację na moim komputerze

WebSockets

- Standard, który pozwala na dwukierunkową, stanową komunikację pomiędzy serwerem, a klientem HTTP
- Obsługiwany przez wszystkie nowoczesne przeglądarki stacjonarne i mobilne
- Poza nawiązaniem połączenia, pomija narzut typowy dla protokołu HTTP

WebSockets

Przykład (klient):

```
GET /czat HTTP/1.1
Host: czat.serwer.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: xIsk230rnci3jdsfij2js==
Sec-WebSocket-Protocol: czat
Sec-WebSocket-Version: 15
Origin: http://serwer.com
```

WebSockets

- Przykład (serwer):

HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HasdOSJds2d244cj452=
Sec-WebSocket-Protocol: czat