

Protokół SSH

Patryk Czarnik

Bezpieczeństwo sieci komputerowych – MSUI 2009/10

Praca na odległość – potrzeby w zakresie bezpieczeństwa

- ▶ Identyfikacja i uwierzytelnienie osoby
- ▶ Uwierzytelnienie serwera
- ▶ Zabezpieczenie przed podszyciem się w trakcie trwania sesji
- ▶ Poufność danych przekazywanych podczas pracy

Potrzeby te są zaspokajane przez protokół SSH

Inne systemy pracy na odległość

- ▶ TELNET
- ▶ RSH
- ▶ X11
- ▶ produkty i protokoły typu „zdalny pulpit”, np. komercyjny Citrix (Windows NT i późniejsze)

Historia

- ▶ SSH-1, 1995, Tatu Ylönen (Fin).
- ▶ SSH-2, 1996, niekompatybilny wstecz, poprawa funkcjonalności (np. wiele sesji na jednym połączeniu) i bezpieczeństwa (m.in. kontrola integralności MAC, wymiana kluczy D-H).
- ▶ W 2006 SSH-2 został standardem IETF (główny dokument: RFC 4251).
- ▶ Pierwsze wersje oryginalnego SSH były otwarte, późniejsze własnościowe (*proprietary*).
- ▶ Od 1999 rozwijana jest otwarta implementacja: OpenSSH, obecnie najbardziej popularna (wg Wikipedii).
- ▶ Liczba implementacji, w tym darmowych, stale rośnie, pojawiają się implementacje na urządzenia mobilne itp.

Ogólne cechy SSH

- ▶ Protokół SSH działa w warstwie sesji modelu ISO/OSI, a w warstwie aplikacji modelu TCP/IP.
- ▶ Opiera się na protokole transportu strumieniowego (TCP).
- ▶ Dodatkowo zapewnia kompresję.
- ▶ Zapewnia tunelowanie sesji terminala.
- ▶ Zapewnia tunelowanie protokołu X11.
- ▶ Zapewnia tunelowanie innych protokołów.
- ▶ Zapewnia negocjację algorytmów kryptograficznych.

Zastosowania SSH

- ▶ Zdalna praca na terminalu tekstowym
 - ▶ także graficznym dzięki tunelowaniu X11.
- ▶ Dostęp do serwerów pozbawionych monitora i klawiatury.
- ▶ Tunelowanie dowolnych połączeń TCP:
 - ▶ zabezpieczenie niezabezpieczonych protokołów (alternatywa dla TLS),
 - ▶ pokonanie firewalli, dostęp do maszyn w wewnętrznej sieci.
- ▶ Bezpieczny transfer plików (przy pomocy dodatkowych protokołów).

Składniki SSH

Protokół składa się z trzech zasadniczych komponentów:

- ▶ Protokołu transportowego
Zapewnia uwierzytelnienie serwera, poufność danych i ich integralność, opcjonalnie obsługuje kompresję.
- ▶ Protokołu uwierzytelnienia użytkownika
Zapewnia uwierzytelnienie klienta.
- ▶ Protokołu połączenia
Multipleksuje różne połączenia.

Protokół transportowy

Funkcjonalność protokołu transportowego (RFC 4253):

- ▶ ustanawianie połączenia,
- ▶ przesyłanie danych (także pozostałych podprotokołów),
- ▶ wymiana kluczy oparta o klucz publiczny,
- ▶ wymiana kluczy Diffiego-Hellmana,
- ▶ ponowna wymiana kluczy,
- ▶ żądanie usługi,
- ▶ inne operacje (rozłączenie, zignorowane dane, debug, zarezerwowane).

Protokół transportowy – ustanawianie połączenia

- ▶ Serwer SSH nasłuchuje (domyślnie) na porcie TCP 22.
- ▶ Na początku następuje wymiana wersji protokołu SSH.
- ▶ Część tego tekstu jest używana w algorytmie Diffiego-Hellmana.
- ▶ Tutaj następuje potwierdzanie tożsamości serwera.
- ▶ Potem następuje dalsza negocjacja, w tym negocjacja kluczy.

Protokół transportowy – przesyłanie danych

Format pakietu z danymi

- ▶ długość całego pakietu (≤ 35000)
- ▶ długość uzupełnienia
- ▶ przesyłane dane (być może skompresowane, dł. ≤ 32768)
- ▶ uzupełnienie (minimalna jednostka transportu: 8 lub dł. bloku szyfrowania; minimalna długość danych 16 lub dł. bloku)
- ▶ MAC (*message authentication code*)

Uwagi

- ▶ „Implementacje mogą pozwalać na przesyłanie dłuższych pakietów”.
- ▶ Jak widać na tym przykładzie – specyfikacja dość zagmatwana. . .

Protokół transportowy – kompresja

- ▶ MAC liczony ze skompresowanych danych
- ▶ Kontekst kompresji jest inicjalizowany przy wymianie kluczy
- ▶ Operacje uaktualnienia kontekstu – po każdym pakiecie
- ▶ Metody kompresji: none, zlib

Protokół transportowy – szyfrowanie

- ▶ Algorytm szyfrowania negocjowany przy wymianie kluczy.
- ▶ Szyfrowaniu podlegają: długość pakietu, długość uzupełnienia, dane, uzupełnienie.
- ▶ Kolejne pakiety tworzą jeden strumień szyfrowany (ważne dla CBC, CFB).
- ▶ Klucze są co najmniej 128-bitowe.
- ▶ Algorytmy szyfrowania: 3DES, BLOWFISH, TWOFISH256, TWOFISH, TWOFISH192, TWOFISH128, AES256, AES192, AES128, SERPENT256, SERPENT192, SERPENT128, ARCFOUR, IDEA, CAST128, NONE.

Protokół transportowy – integralność danych

- ▶ $mac = MAC(klucz, numer_kolejny_pakietu \cdot pakiet)$
- ▶ MAC liczony jest z całego pakietu bez MAC.
- ▶ MAC liczony jest z danych niezaszyfrowanych.
- ▶ Numer kolejny jest zawsze liczony i nie jest zerowany przy renegocjacjach.
- ▶ Numer kolejny pierwszego pakietu to 0.
- ▶ MAC jest przesyłany bez szyfrowania.
- ▶ Stosowane algorytmy: HMAC-SHA1, HMAC-SHA1-96, HMAC-MD5, HMAC-MD5-96 (96 oznacza branie pierwszych 96 bitów wyniku).

Wtrącenie: HMAC

HMAC – algorytm budowy sumy kryptograficznej w oparciu o funkcję haszującą.

$$HMAC_H(K, tekst) = H(K \text{ XOR } opad \cdot H(K \text{ XOR } ipad \cdot tekst))$$

- ▶ K - klucz kryptograficzny używany do uwierzytelnienia;
- ▶ text - uwierzytelniany tekst;
- ▶ \cdot - konkatencja;
- ▶ B - długość bloku funkcji haszującej;
- ▶ ipad - B razy powtórzony bajt 0x36;
- ▶ opad - B razy powtórzony bajt 0x5C;
- ▶ H(...) - algorytm liczenia hasza (czyli np. SHA-1)

Protokół transportowy – negocjacje

- ▶ Na początku każda strona przesyła listę obsługiwanych algorytmów (wszystkie aspekty kryptograficzne i kompresja).
- ▶ Dodatkowo przesyłane jest losowe ciasteczko.
- ▶ Można przyspieszać negocjacje zgadując algorytmy.
- ▶ Potem wysyłany jest komunikat KEXINIT – właściwa negocjacja kluczy.
- ▶ Algorytm Diffiego-Hellmana.
- ▶ Wynikiem jest para: sekret (K) i hash (H).
- ▶ H jest używane jako identyfikator sesji.
- ▶ K i H służą do liczenia wszystkich kluczy oraz wektorów początkowych.
- ▶ Po negocjacji trzeba zatwierdzić wartości specjalnym komunikatem.

Protokół transportowy – algorytmy klucza publicznego

- ▶ Używane do certyfikacji serwera.
- ▶ Zastosowane algorytmy: DSS, RSA, X.509v3 z RSA, X.509v3 z DSS, SPKI z RSA, SPKI z DSS, PGP z RSA i PGP z DSS.

Protokół transportowy – żądanie usługi

- ▶ Obecnie zdefiniowane usługi
 - ▶ uwierzytelnienie użytkownika,
 - ▶ połączenie

Protokół uwierzytelnienia – umiejscowienie

- ▶ RFC 4252.
- ▶ Stosowany po zażądaniu odpowiedniej usługi.
- ▶ Potrzebuje nazwy użytkownika, nazwy usługi oraz nazwy metody.
- ▶ Obsługiwane metody: `none`, `publickey`, `password`, `hostbased`.
- ▶ Dodatkowy standard do wprowadzania nowych metod.
Zastosowanie: `keyboard-interactive`

Protokół uwierzytelnienia – omówienie metod

- ▶ `none` – pusta metoda autentyfikacji, każdy jest wpuszczany lub zatrzymywany, zazwyczaj zablokowana.
- ▶ `publickey` – klient wysyła sygnaturę zaszyfowaną swoim kluczem prywatnym, serwer ją odszyfrowuje odpowiednim kluczem publicznym i wpuszcza, jeśli odszyfrowanie się powiodło.
- ▶ `password` – klient przesyła do serwera zaszyfowane hasło.
- ▶ `hostbased` – klient jest wpuszczany, gdy określony użytkownik loguje się z określonego komputera (pliki `.shosts` w katalogu domowym użytkownika po stronie serwera).

Generyczne uwierzytelnienie za pomocą komunikatów

- ▶ RFC 4256.
- ▶ Standard umożliwia stosowanie metod autentyfikacji klienta innych niż wymienione cztery.
- ▶ Protokół umożliwia autentykację klienta przez serwer za pomocą serii komunikatów.
- ▶ Umożliwia wprowadzenie „nowej” metody autentykacji bez zmian w aplikacji klienta.
- ▶ Obecne główne zastosowanie: metoda `keyboard-interactive`.

Uwierzytelnienie serwera – klucze maszyn

- ▶ Każdy serwer posiada klucz maszyny (co najmniej jeden).
- ▶ Dwa modele wiązania kluczy z maszynami
 - ▶ lokalna baza danych u klienta,
 - ▶ zarządca certyfikatów.

Protokół połączenia – do czego służy

- ▶ RFC 4254.
- ▶ Ustanawianie kanałów komunikacyjnych: terminalowych, X11, tunelowych.
- ▶ Przesyłanie danych.
- ▶ Przekazywanie wartości zmiennych środowiskowych.
- ▶ Kontrola przepływu.
- ▶ Sygnały (jak w UNIX-ie).
- ▶ Kody wyjścia z programów.

Zastosowania SSH – zdalna praca

- ▶ W trybie tekstowym

```
local> ssh -l username remotehost  
remotehost> ls
```

- ▶ Tunelowanie X11

```
local> ssh -Y username@remotehost  
remotehost> firefox
```

Zastosowania SSH – transfer plików

- ▶ Zwykły FTP tunelowany przez SSH.
- ▶ SCP (*secure copy*) – tylko kopiowanie plików. Zależy od systemu (np. dopasowywanie nazw plików robi system operacyjny serwera).
- ▶ SFTP (*SSH file transfer protocol*) – posiada rozbudowaną funkcjonalność (listowanie katalogów, usuwanie plików, zmiana atrybutów) pozwalającą nawet na zamontowanie zdalnego systemu plików (SSHFS). Mniej zależy od systemu.
- ▶ rsync – protokół synchronizacji plików (katalogów), jako warstwy bezpieczeństwa można użyć SSH.

Zastosowania SSH – tunelowanie -L

- ▶ `local> ssh -N -L 8888:www.onet.pl:80`
`username@remotehost &`
`local> links localhost:8888`
- ▶ Otwiera port 8888 na lokalnym komputerze, połączenie z którym tak naprawdę łączy z (w przykładzie) `www.onet.pl:80`.

Zastosowania SSH – tunelowanie -R

- ▶ `local> ssh -N -R 8888:serwis.w.mojej.sieci:80`
`username@remotehost &`
- ▶ Otwiera port 8888 na `remotehost`, połączenie z którym tak naprawdę łączy z (w przykładzie) `serwis.w.mojej.sieci:80`.

Możliwości konfiguracji serwera OpenSSH

- ▶ Wybór dostępnych metod uwierzytelnienia
- ▶ Wybór dostępnych algorytmów kryptograficznych i kompresji
- ▶ Dostępność lub nie tunelowania X11
- ▶ Blokowanie tunelowania określonego typu (domyślnie -R zablokowane dla interfejsów sieciowych innych niż loopback)
- ▶ ...
- ▶ Sekcje reguł dla poszczególnych użytkowników i hostów, z których się łączą