Wprowadzenie do cyberbezpieczeństwa
Temat 1:
Uwierzytelnianie klienta SSH za pomocą kluczy prywatnych

Jakub Stachowicz, 198302
Jan Wiśniewski, 197662

2 czerwca 2025

Spis rzeczy

1 Omówienie SSH i konfiguracja serwera

1.1 Czym jest SSH?

SSH (Secure Shell) to protokół sieciowy, który służy do zdalnego logowania się do innego komputera (serwera) w sposób bezpieczny. Umożliwia zarządzanie systemem operacyjnym, przesyłanie plików oraz wykonywanie poleceń na odległość – wszystko to przy użyciu szyfrowanego połączenia[1, The SSH protocol].

Uwierzytelnianie w SSH może odbywać się na dwa sposoby: za pomocą hasła lub kluczy kryptograficznych. Logowanie hasłem jest proste, ale mniej bezpieczne, ponieważ narażone jest na ataki typu brute-force. Znacznie bezpieczniejszą i częściej stosowaną metodą jest logowanie przy użyciu kluczy SSH, które opiera się na parze klucz prywatny–publiczny. Klucz publiczny umieszczany jest na serwerze, a prywatny pozostaje na komputerze użytkownika, dzięki czemu możliwe jest uwierzytelnienie bez podawania hasła. Metoda ta jest szczególnie przydatna w automatyzacji i pracy z wieloma serwerami[1, Automate with SSH keys, but manage them].

1.2 Algorytmy kryptograficzne w SSH

Temat algorytmów kryptograficznych w protokole SSH dotyczy zarówno szyfrowania samego ruchu sieciowego między klientem a serwerem, jak i logowania i autoryzacji. Niniejsze opracowanie skupia się na algorytmach stosowanych w autoryzacji za pomocą kluczy prywatnych.

W protokole SSH można używać kilku algorytmów generujących parę kluczy, w standardzie zdefiniowano RSA, DSS (DSA) oraz umożliwiono definiowanie innych kluczy[4, 6.6. s. 13]. Najpopularniejsze to RSA, ECDSA i Ed25519 (a dawniej używano też DSA, które dziś jest uważane za przestarzałe). OpenSSH wspiera następujące typy kluczy: DSA, RSA, ECDSA oraz Ed25519[2, User key generation]. Obecnie zaleca się wybór silniejszych algorytmów, np. RSA lub Ed25519 – OpenSSH 7.0 i nowsze domyślnie wyłączają słaby DSA, a od wersji 10.0 wsparcie dla tego algorytmu zostało usunięte w całości[3].

Dalsza część artukułu skupia się na użyciu kluczy RSA, jednak pozostałych algorytmów można użyć w bardzo podobny sposób. Najczęściej – poprzez zmianę parametru `rsa` na np. `ed25519`.

1.3 Konfiguracja z wymuszonym użyciem kluczy

Aby serwer SSH (usługa `sshd`) zezwalał tylko na logowanie kluczem i wyłączał logowanie hasłem, należy zmodyfikować plik konfiguracyjny znajdujący się najczęściej pod ścieżką `/etc/ssh/sshd_config`[5]. Ważne dyrektywy to m.in.:

1. `PubkeyAuthentication yes` – włącza uwierzytelnianie kluczem publicznym (domyślnie zazwyczaj jest włączone).
2. `PasswordAuthentication no` – wyłącza logowanie hasłem. Po jego ustawieniu serwer nie akceptuje haseł SSH.
3. `ChallengeResponseAuthentication no` – wyłącza metody uwierzytelniania oparte na wyzwaniach i odpowiedziach (challenge-response authentication).
4. `AuthorizedKeysFile` – określa lokalizację pliku z kluczami publicznymi (domyślnie dla parametru `%h/.ssh/authorized_keys` będzie to `~/.ssh/authorized_keys`).

Po edycji pliku `sshd_config` należy zapisać zmiany i ponownie uruchomić serwis SSH, np.: `sudo systemctl restart sshd` lub `sudo service sshd reload`. Spowoduje to zastosowanie nowych ustawień.

2 Klucze – generowanie i instalacja

2.1 Generowanie kluczy

Parę kluczy (klucz prywatny i publiczny) generuje użytkownik. Klucz prywatny przechowywany jest po stronie klienta, natomiast klucz publiczny – po stronie serwera.

2.1.1 W systemie Linux – ssh-keygen

W Linuksie (oraz systemach UNIX/macOS) zwykle używa się polecenia `ssh-keygen`. Przykładowo: `ssh-keygen -t rsa -b 4096`. Polecenie `-t rsa -b 4096` tworzy klucz RSA o długości 4096 bitów.

Po uruchomieniu program zapyta o ścieżkę

do pliku (domyślnie `~/.ssh/id_rsa` dla RSA lub `~/.ssh/id_ed25519` dla Ed25519) i opcjonalne hasło (passphrase). Gdy zakończy generowanie, w katalogu `~/.ssh/` powstaną dwa pliki: `id_rsa` (klucz prywatny) i `id_rsa.pub` (klucz publiczny).

2.1.2 W systemie Windows – PuTTYgen

W systemie Windows często korzysta się z PuTTYgen (narzędzie wchodzące w skład pakietu PuTTY). Uruchamiamy PuTTYgen, wybieramy typ klucza (np. RSA lub Ed25519) i długość (np. 2048 bitów), a następnie klikamy przycisk Generate. PuTTYgen poprosi nas o losowe ruchy myszką w obrębie okna – to sposób na zebranie entropii do generacji klucza. Gdy pasek postępu dojdzie do końca, w oknie PuTTYgen pojawi się wygenerowany klucz publiczny (tekst zaczynający się np. `ssh-rsa AAAA...`).

Następnie należy wpisać (dwukrotnie) passphrase chroniący klucz prywatny (zalecane) i zapisać pliki: kliknąć Save public key (np. `mykey.pub`) oraz Save private key (plik `.ppk` dla PuTTY, np. `mykey.ppk`).

W PuTTYgenie można też przekonwertować prywatny klucz do formatu OpenSSH (menu Conversions → Export OpenSSH key) – przydaje się to, gdy chcemy używać tego samego klucza w programach innych niż PuTTY. Plik `.ppk` pozostaje natywnym formatem PuTTY.

2.2 Instalacja kluczy

Po wygenerowaniu kluczy należy je zainstalować – zarówno po stronie klienta, jak i serwera.

2.2.1 Klient SSH Linux

Aby klient logujący się z Linuksa mógł użyć swojego klucza, klucz publiczny klienta należy dodać do pliku `~/.ssh/authorized_keys` konta docelowego na serwerze. Najprościej to zrobić poleceniem `sshcopy-id` – narzędzie automatycznie kopiuje nasz klucz publiczny do odpowiedniego pliku na serwerze. Przykład: `ssh-copy-id user@serwer`.

Powoduje to dodanie zawartości klucza `~/.ssh/id_rsa.pub` (lub innego domyślnego klucza) do `~/.ssh/authorized_keys` na serwerze. Jeśli narzędzie nie jest dostępne, można wykonać ręcznie: najpierw zalogować się hasłem, a potem na serwerze stworzyć katalog `.ssh` i dopisać klucz, np.:

```
mkdir -p ~/.ssh
echo "$(cat ~/.ssh/id_rsa.pub)"
            >> ~/.ssh/authorized_keys
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

Następnie należy zadbać o instalację klucza prywatnego po stronie klienta. W tym celu po wygenerowaniu pary kluczy, plik z kluczem prywatnym (`id_rsa`) powinien zostać skopiowany lub przeniesiony tylko do katalogu domowego użytkownika klienta, w podkatalogu `.ssh`: `mv /path/to/id_rsa ~/.ssh/id_rsa`.

Po dodaniu klucza można przetestować logowanie: `ssh user@serwer` już nie powinno prosić o hasło (chyba że nadaliśmy passphrase do klucza).

2.2.2 Klient SSH Windows (OpenSSH)

Windows (np. Windows 10) posiada wbudowany serwer OpenSSH (lub Win32-OpenSSH). Konfiguracja kluczy jest analogiczna: klucz publiczny wrzucamy do pliku `authorized_keys`. Dla konta użytkownika domyślnie jest to ścieżka: `C:\Users\<user>\.ssh\authorized_keys`.

Dla kont administratora przewidziano specjalny plik w katalogu `C:\ProgramData\ssh\`: `administrators_authorized_keys`. Zawartość tych plików można wprowadzić ręcznie np. przez `scp` lub nawet komendami PowerShell (przykład w dokumentacji Microsoft pokazuje użycie ssh z parametrem, który na zdalnym serwerze tworzy katalog `.ssh` i dopisuje klucz do `authorized_keys`). Ważne jest, aby katalog `.ssh` miał ograniczone prawa (tylko właściciel czy administrator), inaczej OpenSSH może odrzucić klucze.

W tym przypadku plik z kluczem prywatnym powinien trafić analogicznie jak na Linuxie do folderu `.ssh` w foledrze domowym użytkownika: `C:\Users\<user>\.ssh\id_rsa`.

Po umieszczeniu klucza publicznego, logowanie SSH przebiega z użyciem klucza, bez hasła.

2.2.3 Klient SSH PuTTY

W przypadku PuTTY klucz prywatny musi być w formacie PuTTY (`.ppk`). Jeżeli mamy klucz w formacie OpenSSH (np. wygenerowany wcześniej ssh-keygenem), wystarczy go załadować w PuTTYgen (Load private key) i zapisać jako `.ppk` (Save private key). Następnie, aby PuTTY użył klucza, w oknie konfiguracji sesji PuTTY przechodzimy do Connection → SSH → Auth i w polu „Private key file for authentication” wybieramy nasz plik `.ppk`.

W razie potrzeby możemy też użyć Pageanta – agenta kluczy PuTTY – i dodać tam klucz. Klucz publiczny PuTTY (tekst z pola „Public key for pasting...” w PuTTYgen) należy skopiować na serwer do `~/.ssh/authorized_keys` tak samo, jak w poprzednich metodach. Po dodaniu klucza do `authorized_keys` należy skonfigurować PuTTY wskazując wygenerowany prywatny plik `.ppk` i przetestować połączenie.

3 Zastosowania SSH

4 Bibliografia

Artykuły

[1] Tatu Ylonen, *What is SSH (Secure Shell)?*, `https://www.ssh.com/academy/ssh`, [dostęp 24.05.2025].

[2] Microsoft Learn, *Key-based authentication in OpenSSH for Windows*, `https://learn.microsoft.com/en-us/windows-server/administration/openssh/openssh_keymanagement` [dostęp 24.05.2025].

[3] OpenSSH Release Notes, *OpenSSH 10.0*, `https://www.openssh.com/txt/release-10.0`, [dostęp 24.05.2025].

Dokumentacje techniczne

[4] Chris Lonvick, Tatu Ylonen, Styczeń 2006, *The Secure Shell (SSH) Transport Layer Protocol*, `https://www.rfc-editor.org/rfc/rfc4253`, [dostęp 24.05.2025].

[5] OpenBSD manual page server, *ssh_config*, `https://man.openbsd.org/ssh_config`, [dostęp 24.05.2025].