Bezpieczne SSH

Marcin Ulikowski

SSH jest popularnym standardem protokołów oferującym bezpieczne, szyfrowane połączenia. Znajduje wiele zastosowań, od terminalowego łączenia ze zdalnym komputerem przez transfer plików do tunelowania połączeń.

autorzy@lpmagazine.org

Został zaprojektowany jako następca wysłużonych protokołów takich jak Telnet, FTP czy RSH które wszelkie dane, także hasła, przesyłały w sposób jawny. Posiada bardzo istotną cechę – umiejętność potwierdzenia tożsamości komputera z którym nawiązujemy połączenie. Jeśli weryfikacja przebiegnie pomyślnie, nie ma możliwości aby ktokolwiek mógł odczytać lub zmodyfikować zawartość transmisji. Jednak pomimo szyfrowanego połączenia możliwe jest podsłuchiwanie sesji, chociaż proces ten ma inny charakter niż w przypadku starszych protokołów. Atak ten nosi nazwę *Man in the middle* (ang. człowiek po środku), czyli atak z pośrednikiem.

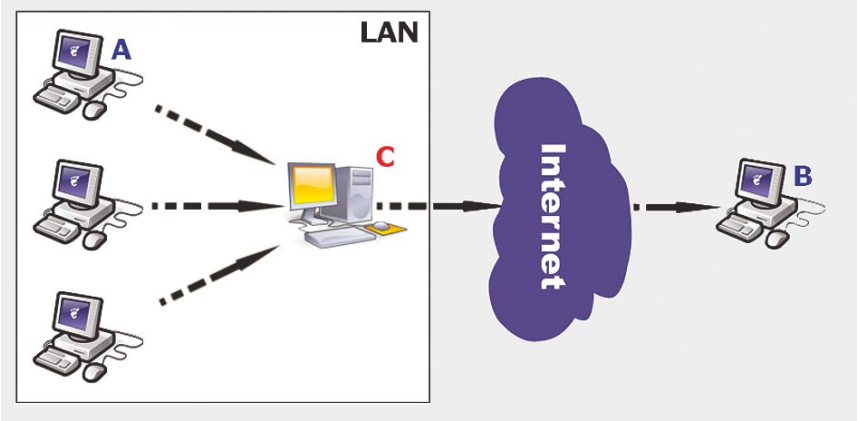
Schemat ataku

Atak z pośrednikiem polega na czytaniu i modyfikowaniu treści połączenia pomiędzy dwoma stronami bez ich wiedzy. Atakujący pełni rolę pośrednika w komunikacji między dwoma komputerami, podobnie jak to robią serwery proxy. Jednak istotna różnica polega na tym, że nikt nie wie o istnieniu pośrednika. W sposób niewidoczny aktywnie monitoruje, przechwytuje i kontroluje komunikację. Osoba po drugiej stronie połączenia myśli, że komunikuje się z inną osobą, podczas gdy w rzeczywistości komunikuje się z napastnikiem.

Wyobraźmy sobie sytuację w której komputer A znajdujący się w sieci lokalnej chce nawiązać połączenie zdalne z komputerem B. Po drodze stoi komputer C, czyli bramka NAT pod kontrolą atakującego. Oznacza to, że wszystkie dane wysyłane w stronę Internetu są przekazywane za pośrednictwem komputera C. Komputer ten może imitować serwer usługi z którą chcemy się połączyć. Jeśli atak się powiedzie, komputer A będzie bezpośrednio połączony z maszyną napastnika myśląc, że jest połączony z właściwym serwerem. Napastnik realizując jednocześnie połączenie z komputerem A i serwerem docelowym B będzie miał dostęp do wszystkich przesyłanych informacji, łącznie z hasłem dostępu. Po zdobyciu hasła może zerwać połączenie lub wciąż pełnić rolę przekaźnika, jeśli chce aby komputer A pozostał nieświadomy faktu, że stał się ofiarą.

SSH i klucze publiczne

Protokół SSH standardowo udostępnia możliwość obrony przed atakiem tego rodzaju. Klient, który realizuje szyfro-



Rysunek 1. Schemat połączenia, w którym komputer C kontrolowany jest przez napastnika

wane połączenie za każdym razem sprawdza tożsamość serwera. Weryfikacja opiera się na sprawdzeniu zgodności klucza publicznego danego serwera. Klucz serwera zapisywany jest na stale w bazie klienta i może się tam znaleźć na trzy sposoby:

- akceptacja klucza przesyłanego przez serwer podczas pierwszego połączenia (najczęściej stosowane rozwiązanie),
- administrator lub inna kompetentna osoba udostępnia klucz publiczny serwera, na przykład na stronie internetowej (zalecane, ale rzadko spotykane rozwiązanie),
- znajoma osoba (darzymy ją zaufaniem), która posiada dostęp do serwera, udostępnia nam wcześniej otrzymany klucz publiczny.

Pierwsze rozwiązanie jest najczęściej stosowane. Łącząc się po raz pierwszy z danym serwerem, otrzymujemy informację, iż nie posiadamy jeszcze takiego klucza publicznego w swojej bazie. W przypadku klienta *OpenSSH* będzie to komunikat podobny do tego na Rysunku 2. Popularna implementacja SSH dla Windows, klient *PuTTY*, wyświetli komunikat zbliżony do przedstawionego na Rysunku 3. Od nas zależy, czy na pytanie o przyjęcie nowego klucza odpowiemy twierdząco. Jeśli zdecydujemy się zapisać klucz, połączenie zostanie zabezpieczone. Teraz użytkownik jest najsłabszym ogniwem protokołu. Gdy dochodzi do ataku, czyli między nami i serwerem stoi osoba trzecia, otrzymujemy od serwera (w rzeczywistości od napastnika) inny klucz publiczny. W tym momencie klient SSH informuje o tym fakcie oraz sugeruje wystąpienie potencjalnego ataku. Częstym błędem popełnianym przez użytkowników jest nie czytanie komunikatów tego typu lub ich bagatelizowanie, co najczęściej wynika z nieświadomości istniejącego zagrożenia. Powodzenie ataku zależy więc od użytkownika. Dlatego podczas nawiązywania połączenia z serwerem, nigdy nie wolno akceptować zmienionego klucza publicznego chyba, że mamy pewność co do jego pochodzenia. Bardzo dobrym zwyczajem jest także stosowanie osobnych haseł dla posiadanych kont, dzięki czemu ewentualne włamanie nie pociągnie za sobą kolejnych.

Dlaczego klucz publiczny serwera ulega zmianie?

Zmiana klucza publicznego nie zawsze oznacza próbę włamania na nasze konto. Dlatego jeśli otrzymujemy komunikaty takie jak na Rysunku 4 i Rysunku 5, nie wpadajmy w panikę. Najczęściej przyczyną ich wystąpienia jest:

- zmiana wersji protokołu negocjowanego przez serwer ze starszej SSHv1 (nie zalecanej do stosowania) na nowszą SSHv2,
- aktualizacja demona SSH lub całkowita aktualizacja systemu w wyniku której nie został zachowany (przeoczenie ze strony administratora) wcześniej używany klucz publiczny,
- zmiana adresu IP lub nazwy serwera.

Najlepszym rozwiązaniem jest wstrzymanie się z akceptacją zmienionego klucza i kontakt z osobą zarządzającą serwerem w celu wyjaśnienia przyczyny zmiany klucza oraz uzyskanie nowego, właściwego.

```
(elceef@osiris ~)$ ssh elceef@[illegible].pl
The authenticity of host '[illegible].pl ([illegible].204.60.225)' can't be established.
RSA key fingerprint is 1e:d6:0d:f2:c6:eb:cf:ff:e4:cd:e3:e1:b3:fd:46:ec.
Are you sure you want to continue connecting (yes/no)?
```

Rysunek 2. Informacja o nieznanym kluczu publicznym (OpenSSH)

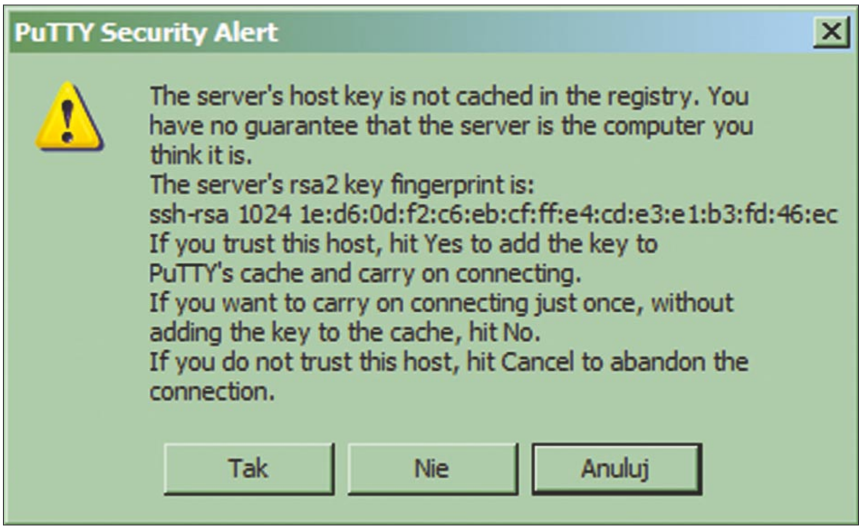
Gdy jest za późno

Co zrobić, gdy zaakceptowaliśmy nieznany klucz i zaczynamy podejrzewać, że staliśmy się ofiarą ataku *Man in the middle*? Możemy to sprawdzić w bardzo prosty sposób na własną rękę. Potrzebny będzie tzw. fingerprint (ang. odcisk palca) klucza oraz informacja jakim algorytmem został utworzony. W naszym przykładzie (Rysunek 4) jest to algorytm RSA. Odcisk palca to 128-bitowa liczba zapisana w postaci szesnastkowej, gdzie oktety oddzielone są dwukropkami. Musimy zalogować się na serwer w ce-

Baza kluczy publicznych

Baza znanych serwerów jest również źródłem ciekawych informacji dla włamywacza. W przypadku *OpenSSH* plik *known_hosts* zawiera adresy IP (najczęściej także pełne nazwy) serwerów oraz ich klucze publiczne. Dane te nie są zabezpieczane, ponieważ klucze publiczne mogą być rozpowszechniane bez obaw o naruszenie bezpieczeństwa serwera. Jednak duża ilość takich danych zebrana w jednym miejscu stanowi już źródło pewnych informacji. Tworzy się lista serwerów SSH na których użytkownik posiada konto (jak pokazuje praktyka, najczęściej z takim samym hasłem). Bazy kluczy publicznych to doskonały materiał dla włamywacza. Analiza nieuprawnionych logowań na konta użytkowników pokazuje, że włamywacze uzyskując dostęp do plików *known_hosts* zyskują dalszy dostęp do innych kont. Główną przyczyną są takie same hasła stosowane na wszystkich serwerach oraz niezaszyfrowane klucze RSA (umożliwiają zalogowanie się na serwer bez podania hasła).

Rozwiązanie problemu jest dosyć proste i zostało wbudowane w *OpenSSH* 4.0 oraz nowsze wersje (opcja `HashKnownHosts` w pliku konfiguracyjnym). Polega na przechowywaniu adresu w postaci skrótu (*hash*), zamiast w postaci jawnej. W przypadku gdy klient łączy się z serwerem, skrót umożliwia łatwe sprawdzenie, czy adres znajduje się w pliku. Natomiast nie jest możliwe działanie odwrotne, czyli odzyskanie adresu serwera ze skrótu. Dzięki temu włamywacz nie dowie się z jakich serwerów korzysta użytkownik.



Rysunek 3. Informacja o nieznanym kluczu publicznym (PuTTY)

lu sprawdzenia odcisku palca. Wykorzystamy polecenie *ssh-keygen* oraz plik z kluczem publicznym serwera. W naszym przypadku będzie to plik *ssh_host_rsa_key.pub*, ponieważ odcisk palca został wygenerowany przy użyciu algorytmu RSA (SSHv2). Dostęp do tego pliku powinien mieć każdy użytkownik serwera.

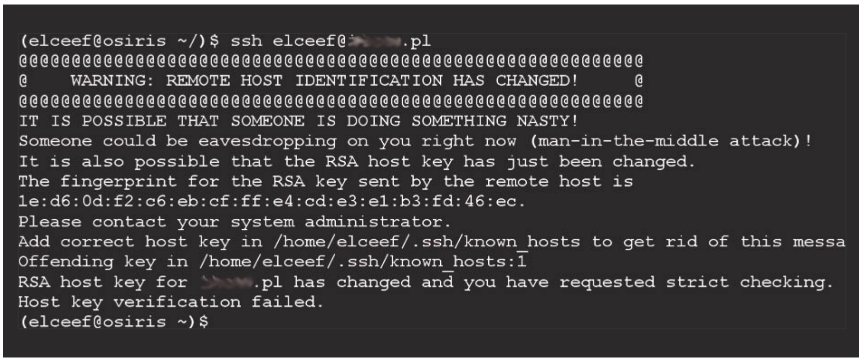
```
$ ssh-keygen -l -f /etc/ssh/
  ssh_host_rsa_key
1024 1e:d6:0d:f2:c6:eb:cf:ff:
        e4:cd:e3:e1:b3:fd:46:ec
```

W naszym przykładzie odcisk palca z Rysunku 2 i zwrócony przez polecenie *ssh-keygen* są identyczne, więc nie mamy powodu do obaw. Klucz został zmieniony w wyniku działań administratora serwera. Jeśli odciski palca są różne, to znaczy, że prawdopodobnie staliśmy się ofiarą ataku. W takim przypadku należy natychmiast zmienić hasło dostępu do konta oraz skontaktować się z administratorem systemu w celu wyjaśnienia sytuacji.

Warto zapamiętać, że najpewniejszym źródłem informacji jest administrator. Napastnik ma pełną kontrolę nad treścią naszego połączenia. Możliwe jest więc, że informacje zwracane przez program *ssh-keygen* zostaną podczas transmisji podmienione na fałszywe, aby uśpić naszą czujność. Dlatego zawsze w takich sytuacjach warto skontaktować się z osobą zarządzającą daną usługą na serwerze.

Podsumowanie

Przeprowadzenie ataku *Man in the middle* nie jest trudne. Istnieją gotowe narzędzia, dzięki którym nawet średnio-zaawansowany użytkownik przy odrobinie szczęścia może podsłuchać nasze połączenie. Należy jednak pamiętać, że będzie to możliwe tylko wtedy, gdy mu na to pozwolimy.



Rysunek 4. Linuksowy klient SSH informuje o zmianie klucza publicznego

Rodzaje kluczy

Wykorzystywane są dwie wersje protokołu: SSHv1 oraz SSHv2. Starsza wersja SSHv1 wykorzystuje algorytm RSA, natomiast nowsza SSHv2 polega na algorytmach RSA i DSA. Możemy więc korzystać z trzech rodzajów kluczy. W pliku konfiguracyjnym */etc/ssh/sshd_config* określane są on kolejno jako `rsa1`, `rsa`, `dsa`. Podczas instalacji serwera SSH generowane są trzy różne klucze oparte na wymienionych algorytmach. Klucze prywatne są później przechowywane w plikach: *ssh_host_key* (rsa1), *ssh_host_rsa_key* (rsa) oraz *ssh_host_dsa_key* (dsa). Dostęp do nich powinien mieć tylko administrator. Pliki o takich samych nazwach, ale z rozszerzeniem *.pub* zawierają tylko klucze publiczne i dostęp do nich powinien być swobodny. Do generowania kluczy prywatnych oraz publicznych (nie tylko) służy polecenie *ssh-keygen*.

Rysunek 5. Program PuTTY ostrzega o potencjalnym ataku