DANIEL SUCHOCKI

Port knocking

Stopień trudności

Zabezpieczanie sieci komputerowych przed niepowołanym dostępem z zewnątrz to temat bardzo rozległy i pełen rozmaitych koncepcji, z których niektóre wzajemnie się wykluczają. Są też rozwiązania uniwersalne, a – co więcej – beznakładowe, które pozwalają utrudnić nieuprawniony dostęp. Zalicza się do nich Port knocking.

Z ARTYKUŁU DOWIESZ SIĘ

co to jest technika port knockingu,

jakie są podstawy funkcjonowania port knockingu,

jak skonfigurować proste zabezpieczenie oparte o port knocking i iptables.

CO POWINIENEŚ WIEDZIEĆ

powinieneś mieć podstawowe pojęcie na temat protokołu TCP/IP,

powinieneś swobodnie poruszać się w środowisku systemu Linux,

powinieneś mieć podstawową wiedzę na temat kompilacji oprogramowania w systemie Linux.

Jednym z fundamentalnych założeń zachowania poufności systemów informatycznych jest uniemożliwienie dostępu do przechowywanych informacji. Ten podstawowy cel wszystkich zasad bezpieczeństwa opiera się w dużej mierze o zaufanie do stosowanych rozwiązań i rzadko wykracza poza ogólnie przyjęte schematy zabezpieczeń. To z kolei ułatwia działanie włamywaczom (krakerom), ponieważ techniki włamań z jednej strony starają się odnaleźć możliwe podatności systemu i zastosowanych w nim rozwiązań, a z drugiej – zakładają pewien rodzaj standaryzacji. Włamanie do systemu na ogół ma na celu przejęcie lub pozyskanie prawa dostępu do tego systemu, a następnie wydobycie lub zmianę określonych informacji. Coraz częściej takie włamania prowadzone są na zlecenie, a jedyne sensowne wnioski wynikają z analizy powłamaniowej lub stosowanej w ramach informatyki śledczej. Co ciekawe, statystyki tych analiz wskazują, że znaczna część włamań do systemów opiera się o powszechnie dostępne narzędzia i techniki opisane na forach internetowych!

Kluczem do dobrego bezpieczeństwa systemu informatycznego są więc nie tylko zastosowane środki (oprogramowanie, systemy), ale również niestandardowe podejście, wymagające większego zaangażowania i wiedzy ze strony potencjalnego włamywacza. Oczywiste metody to stosowanie wielopoziomowych zabezpieczeń i złożonych haseł, ale warto rozważyć wykorzystanie mniej powszechnych rozwiązań.

Port knocking w praktyce

Dobry i sumienny administrator stale poszukuje nowych sposobów na zabezpieczenie swoich serwerów. Najbardziej powszechne jest stosowanie haseł jako metody uwierzytelniania. Silne i skomplikowane hasła dają poczucie bezpieczeństwa. Inną metodą jest używanie metod kryptograficznych opartych na kluczach prywatnych i publicznych. Na nic jednak nie zdadzą się i silne hasła, i specjalnie przygotowane klucze, jeśli napastnik wykryje podatność w oprogramowaniu i wykorzysta ją poprzez np. otwarty port naszego serwera. Dlatego najlepszym rozwiązaniem, które nasuwa się przy rozważaniu kwestii ataków na serwery internetowe, jest zamknięcie wszystkich portów i **odcięcie się od świata**. Wyjątkiem są tutaj oczywiście serwery udostępniające usługi HTTP lub SMTP, które z założenia powinny przyjmować połączenia od każdego i skądkolwiek.

Co jednak ma począć administrator, który musi mieć **kontakt** ze swoimi maszynami, a standardowy port, na którym działa powszechnie znana metoda połączenia zdalnego (tcp/22), jest zamknięty? Odpowiedzią jest właśnie technika **port knockingu**, umożliwiająca zdalnemu użytkownikowi wysłanie żądania otwarcia określonego portu. Aby użytkownik mógł uzyskać dostęp, musi w określonej kolejności wysłać sekwencję pakietów

na zamknięty port serwera. Mimo, iż takie połączenia są nieudane, to zdefiniowana seria pakietów aktywuje demona, który tak modyfikuję regułę zapory sieciowej systemu, aby zezwolić na połączenie z określonego adresu IP – jeśli użytkownik udowodni, że ma do tego prawo. Pojawia się kolejne pytanie: jak wysłać informacje na port, który jest zamknięty i nie nasłuchuje na nim żadna usługa? Wbrew powszechnym przekonaniom jest to możliwe, a port nie musi być otwarty, aby przesłać do niego dane. Nawet wtedy, gdy połączenia są odrzucane przez zapory sieciowe, to wszelkiego rodzaju próby przedostania się wchodzących i wychodzących pakietów są monitorowane oraz logowane. Dzięki temu w wysłanym pakiecie można ukryć sekwencję, która spowoduje wykonanie skojarzonych z nią zadań.

Do dzieła…

Jest wiele implementacji **port knockingu**, a na przełomie lat ta metoda coraz bardziej się rozwija, ukazując administratorom nowe sposoby i drogi, którymi mogą podążać, tworząc własne pomysły. Niektóre odmiany **port knockingu** przedstawia Tabela 1.

Na potrzeby tego artykułu zostanie omówiona metoda **port konckingu** z wykorzystaniem standardowego narzędzia do filtracji pakietów oraz programu Bruce'a Warda o nazwie Doorman. Jego instalacja i konfiguracja nie jest zbyt skomplikowana, a po poprawnym skonfigurowaniu staje się on bardzo przydatnym narzędziem, stanowiącym doskonały dodatkowy pierścień zabezpieczeń serwera internetowego. Doorman w celu **puknięcia** wykorzystuje pojedynczy pakiet UDP, który zawiera ciąg uwierzytelniający. **Puknięcie** składa się z kombinacji numeru portu, do którego chcemy uzyskać dostęp, identyfikatora użytkownika oraz liczby losowej. Po odebraniu przez Doormana pakietu zostaje on zweryfikowany pod kątem zawartości plików konfiguracyjnych znajdujących się na serwerze i, jeśli wszystko się zgadza, Doorman dopuszcza użytkownika do żądanego portu. Program jest również wyposażony w środki umożliwiające uniknięcie ataku powtórzeniowego, a jego największą zaletą jest kompatybilność z większością firewalli stosowanych w systemach UNIX (dla starszych jąder ipchains, dla nowszych iptables oraz ipfw, który występuje w systemach np. FreeBSD).

Zanim **port knocking** zacznie poprawnie funkcjonować na serwerze, musimy odpowiednio skonfigurować firewall. Firewall w systemach linuksowych to nic innego jak filtr pakietów, dostępny we wszystkich systemach operacyjnych. Stanowi on podstawę ograniczeń dostępu do lokalnych usług, co świetnie przekłada się na możliwość rozpoznania i zablokowania prób ataku. Filtr pakietów to zbiór reguł określających, co system powinien zrobić z pakietem przychodzącym z sieci (lub do niej wychodzącym). Filtr jest sterowany zbiorem reguł, których podstawowymi elementami są wzorce oraz akcje, określające, co zrobić z pakietem pasującym do danej reguły. W skład wzorca mogą wchodzić cechy charakterystyczne dla protokołu IP, takie jak

Listing 1. Podstawowe reguły przygotowane na potrzeby port knockingu

```
#!/bin/sh

iptables=/sbin/iptables

$iptables -F
$iptables -t nat -F

$iptables -P INPUT DROP
$iptables -P FORWARD DROP
$iptables -P OUTPUT ACCEPT

$iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Tabela 1. Wybrane rodzaje port knockingu

Nazwa implementacji portknockingu:	Język w jakim został napisany:	Protokół, który jest wykorzystywany:	Dostęp:
Doorman	C	UDP	Poprzez reguły zapory
			Poprzez trwały stan
fwknop	C	TCP/UDP/ICMP	Poprzez reguły zapory
			Poprzez dowolną komendę
SAd00r	C	TCP/UDP/ICMP	Poprzez dowolną komendę
COK	Java	TC/UDP	Poprzez reguły zapory
			Poprzez dowolną komendę
			Poprzez trwały stan
knockd	C	TC/UDP	Poprzez reguły zapory
			Poprzez dowolną komendę

```
# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere            state RELATED,ESTABLISHED

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Rysunek 1. Widok reguł zapory iptables

adres źródłowy i docelowy pakietu, numery portów protokołów TCP i UDP, rozmaite flagi, typ komunikatu ICMP i inne. Przetwarzanie reguł odbywa się dla każdego pakietu przychodzącego lub wychodzącego z danego węzła. Pakiet, który pasuje do określonego w danej regułce wzorca, jest traktowany zgodnie z przypisaną do niego akcją. Zwykle ogranicza się ona do przepuszczenia lub zablokowania pakietu. W systemach linuksowych do wyboru są trzy rodzaje filtrów, różne dla poszczególnych numerów serii jądra systemu. Dla jąder z serii 2.0.x jest to **ipwf**, dla jąder z serii 2.2.x jest to **ipchains**, natomiast dla jąder z serii 2.4.x – **iptables**. Zasada działania takiego filtra jest dość prosta. IPtables posiada trzy domyślne zestawy reguł `INPUT`, `FORWARD` i `OUTPUT` oraz szereg celów, które określają, co należy zrobić z pakietem pasującym do danej regułki. Najczęściej stosowane reguły to `ACCEPT`, `DROP`, czyli zablokowanie pakietu bez powiadomienia nadawcy, oraz `REJECT` – czyli zablokowanie wraz ze zwróceniem komunikatu ICMP. Jednak przy połączeniu iptables wraz z **port knockingiem** nie trzeba budować zaawansowanych reguł. Dobrym pomysłem jest utworzenie pliku o nazwie np. **firewall**, umieszczenie w nim potrzebnych reguł i dołączenie ścieżki do niego do listy usług startujących wraz z systemem. Listing 1 przedstawia plik z podstawowymi regułami, przygotowany na potrzeby współpracy z **port knockingiem**. Dwa pierwsze polecenia kasują nam istniejące reguły zapory sieciowej.

```
$iptables -F
$iptables -t nat -F
```

Trzy następne ustawiają nam kolejno: odrzucanie połączeń przychodzących, odrzucanie połączeń przechodzących oraz akceptowanie połączeń wychodzących:

```
$iptables -P INPUT DROP
$iptables -P FORWARD DROP
$iptables -P OUTPUT ACCEPT
```

Ostatnia linijka umożliwia utrzymanie już istniejących (nawiązanych) połączeń:

```
$iptables -A INPUT -m state --state
   ESTABLISHED,RELATED -j ACCEPT
```

Aby zobaczyć, jak układają się reguły `iptables`, należy uruchomić wcześniej przygotowany plik. Zakładając, iż umieściliśmy go w katalogu **/etc/init.d/firewall**, wydajemy komendę:

```
# /etc/init.d/firewall start
```

Wydając polecenie:

```
# iptables -L
```

Uzyskamy widok reguł zapory iptables (Rysunek 1).

Mając już gotowe reguły firewalla, można przystąpić do konfiguracji Doormana. Po pobraniu źródeł programu należy rozpakować plik w dowolnym katalogu. Zanim jednak rozpoczniemy proces konfiguracji, należy upewnić się, czy nasz system wyposażony jest w oprogramowanie: libpcap, flex oraz lsof, gdyż jest ono wymagane do poprawnej kompilacji programu Doorman. Po poprawnym **uzbrojeniu** naszego systemu w odpowiednie pliki możemy przystąpić do kompilacji, wchodząc we wcześniej rozpakowany katalog i wydając w nim polecenia `# ./configure && make` oraz (z konta użytkownika root) `# make install`.

Po wydaniu tych komend w katalogu **/usr/local/etc/doormand** powstaną pliki konfiguracyjne, w katalogu **/usr/local/bin** znajdziemy plik **knock** potrzebny do **pukania**, a w katalogu **/usr/local/sbin** pojawi się program Doormand.

Kolejne działania należy rozpocząć od stworzenia odpowiednich plików konfiguracyjnych na potrzeby naszego systemu. Główny plik konfiguracji Doormana, zawierający m.in. informacje o numerze portu, na którym nasłuchuje aplikacja, interfejsie, do którego będzie się ona odwoływała, czasie, po którym porty mają ponownie się zamknąć oraz ścieżce dostępu do innych plików konfiguracyjnych, znajduje się w lokalizacji **/usr/local/etc/doorman/doorman.cf**. Domyślnie program nasłuchuje na porcie 1001, ale dobrym pomysłem będzie jego zmiana na inny, gdyż zwiększy to odporność na włamania realizowane przez osoby, które poznały naszą metodę zabezpieczeń. Przykładowy plik konfiguracyjny przedstawia Listing 2.

Przykładowy plik konfiguracyjny przygotowany jest pod pierwszy interfejs sieciowy **eth0**, jednak serwery linuksowe posiadają co najmniej dwa interfejsy do ruchu sieciowego wewnętrznego oraz zewnętrznego. Trzeba więc samodzielnie przygotować drugi plik konfiguracyjny, zmieniając jedynie nazwę interfejsu sieciowego np. **eth1**, umieścić go w tym samym katalogu i nazwać np. **/usr/local/etc/doorman/doorman.cf.eth1**. Kolejnym plikiem, na który należy zwrócić uwagę, jest **/usr/local/etc/doorman/questlist**, który

Listing 2. Przykładowy plik konfiguracyjny

```
#
#  'doormand.cf'
#  Sample configuration file for the Doorman Daemon, "doormand".
#
#
interface            eth0
port                 1001
waitfor              10
connection_delay_1   100000  # 1/10th second (delay is in microseconds)
connection_delay_2   2
logfile             /var/log/messages
loglevel             debug
pidfile             /var/run/doormand.pid
guestlist            /usr/local/etc/doormand/guestlist
firewall-add        /usr/local/etc/doormand/iptables_add
firewall-del        /usr/local/etc/doormand/iptables_delete
hash-archive         /var/doormand.hash-archive
hash-archive-size    100000
```

Listing 3. Plik konfiguracyjny przechowujący dane o autoryzowanym użytkowniku

```
tomek      niemamoc      21 22          0.0.0.0/0
adam       CeZaR            22       87.234.67.54
slawek     S023210          21     192.168.0.0/24
```

przechowuje nazwy użytkowników, haseł, numery portów, które dany użytkownik ma prawo otwierać oraz sieć, z której może **pukać** do serwera. Przykładowy plik przedstawia Listing 3.

Możemy zatem operować na wielu różnych portach, jak również przypisywać do nich konkretne adresy IP oraz całe sieci. Tak przygotowany Doorman jest gotowy do **odpalenia** i ochrony naszego serwera.

Aby wystartować Doormana, należy wydać polecenie `/usr/local/sbin/doormand -D`, co spowoduje uruchomienie usługi wraz z debugowaniem wiadomości na monitorze. Musimy także pamiętać o uruchomieniu Doormana nasłuchującego na drugim interfejsie, tak aby administrator miał prawo logować się zarówno z zewnątrz, jak i z wewnątrz sieci. Do wyspecyfikowania innych plików konfiguracyjnych niż standardowy służy opcja `-f (/usr/local/sbin/doormand -D -f /usr/local/etc/doormand/doormand.cf.eth.1`). Tak uruchomiony Doorman będzie logował cały ruch na ekranie konsoli – jest to bardzo dobra metoda, gdyż dzięki niej od razu widać, kto, kiedy i skąd próbuje się dostać na serwer. Z drugiej strony – jak powszechnie wiadomo – administrator jest osobą bardzo zabieganą i niemającą czasu na przeglądanie logów migających na konsoli. Dodatkowo nie będzie bardzo estetycznym włączanie Doormana za każdym razem przy restarcie serwera. Dlatego dobrym wyjściem jest stworzenie własnych skryptów startowych i dopisanie ich do reguł startujących wraz z systemem operacyjnym. Przykładowe pliki konfiguracyjne dla interfejsu **eth0** i **eth1** przedstawiają kolejno Listing 4. oraz Listing 5.

Tak przygotowane skrypty należy uruchomić prostym poleceniem, przyjmując, że zostały skopiowane do katalogu **/etc/init.d**:

```
# /etc/init.d/doormand_eth0 start
# /etc/init.d/doormand_eth1 start
```

Przypisując skrypty do reguł startujących wraz z systemem, nie będziemy musieli się martwić, czy Doorman został uruchomiony, czy też nie. Po uruchomieniu skryptów startowych Doorman jest gotowy i, nasłuchując, czeka na połączenie.

Nadszedł czas na konfigurację klienta, z którego będziemy się łączyć z serwerem. Na początek możemy skompilować i zainstalować program – w taki sam sposób, jak odbywało się to na serwerze. Jednak tym razem jedynym plikiem, który jest nam potrzebny, jest **/usr/local/bin/knock**. Tak naprawdę klient jest już przygotowany do **pukania** na serwer, a co za tym idzie – do otwierania portów i dostania się do konkretnych usług. Można tego dokonać wydając polecenie:

```
# /usr/local/bin/knock -g tomek
   -p 1001 -s niemamoc <IP_serwera>
```

Po wysłaniu pakietu można zobaczyć, że Doorman wykrył **puknięcie** i za pomocą skryptu `/usr/local/etc/doormand/iptables _ add` dynamicznie zmienił zasady zapory sieciowej (Rysunek 2).

Teraz administrator ma 10 sekund (opcja **waitfor** w pliku **doormand.cf**)

Listing 4. Przykładowe pliki konfiguracyjne interfejsu eth0

```
#!/bin/sh

NAME=doormand_eth0
DAEMON=/usr/local/sbin/doormand
PIDFILE=/var/run/doormand_eth0.pid

case $1 in
    start)
    echo "Starting doormand eth0 ... done"
    start-stop-daemon --start --exec $DAEMON --pidfile $PIDFILE -- -p $PIDFILE
    echo "Lisining on eth0"
    ;;

    stop)
    echo "Stopping doormand eth0 ..."
    start-stop-daemon --stop --exec $DAEMON --pidfile $PIDFILE -- -p $PIDFILE
    echo "done ..."
    ;;

    restart)
    ;;
esac
```

Listing 5. Przykładowe pliki konfiguracyjne interfejsu eth1

```
#!/bin/sh
NAME=doormand_eth1
DAEMON=/usr/local/sbin/doormand
PIDFILE=/var/run/doormand_eth1.pid
CONF=/usr/local/etc/doormand/doormand.cf.eth1

case $1 in

     start)
     echo "Starting doormand eth1 ...done"
     start-stop-daemon --start --exec $DAEMON --pidfile $PIDFILE -- -p $PIDFILE -f
                       $CONF
     echo "Lisining on eth1"
     ;;

     stop)
     echo "Stoping doormand eth1 ..."
     start-stop-daemon --stop --exec $DAEMON --pidfile $PIDFILE -- -p $PIDFILE
     echo "done ..."
     ;;

     restart)
     ;;

esac
```

na próbę połączenia z portem, po czym zapora firewall wróci do swojego początkowego stanu, zamykając port. Jeśli administrator połączy się z usługą SSH, nie zostanie ona rozłączona dzięki linijce:

```
$iptables -A INPUT -m state --state
   ESTABLISHED,RELATED -j ACCEPT
```

w pliku konfiguracji filtra pakietów, która powoduje utrzymanie już nawiązanego połączenia.

Ten sposób jest o tyle uciążliwy, że za każdym razem użytkownik musi pamiętać swój login, hasło oraz adres IP serwera, na który chce się dostać. Jest na to rada. Dobrym i zarazem bardzo bezpiecznym pomysłem jest stworzenie skryptu, który automatycznie podczas nawiązywania połączenia z Doormanem będzie otwierał i łączył się z portem SSH, czekając jedynie na hasło użytkownika do usługi SSH. Żeby tego dokonać, potrzebne będą dwa pliki. Jeden, główny, ukryty plik w katalogu **/root** (dzięki temu nikt poza administratorem nie będzie w stanie edytować pliku, aby podejrzeć lub zmienić jego zawartość) oraz drugi, który będzie specjalnie przygotowany dla administratora. Pierwszy z plików, **.knockcf**, umieszczony w katalogu **/root**, będzie zawierał nazwę użytkownika, numer portu, na którym nasłuchuje Doorman, hasło, oraz komendę, która będzie się uruchamiała, jeśli wszystkie wcześniejsze dane będą się zgadzały. Listing 6 przedstawia taką właśnie konfigurację.

Natomiast drugi plik (zwany kluczem) administrator może przenosić np. na pendrivie, gdyż nie jest on plikiem kluczowym dla bezpieczeństwa. Zawiera jedynie nazwę programu knock, adres IP serwera oraz port, który należy otworzyć. Dla przeciętnego włamywacza te informacje nie będą znaczące. Konfigurację pliku-klucza przedstawiono na Listingu 7.

Teraz wystarczy, że administrator wyda komendę `# ./klucz` i nastąpi połączenie z serwerem. Doorman sprawdzi poprawność loginu i hasła i na 10 sekund włączy port SSH, po czym dzięki plikowi **.knockcf** od razu połączy użytkownika z usługą, pytając o hasło do SSH. Po wylogowaniu się administratora Doorman zauważy ten fakt i wykasuje udoskonaloną regułę zapory. Dzięki takiej konfiguracji tylko użytkownik posiadający klucz dostępu i **pukający** wyłącznie ze swojego systemu jest w stanie pomyślnie połączyć się z serwerem. Istnieje również plik **knock** wykorzystywany w systemach Windows. Tutaj także możemy wykorzystać plik wsadowy do **puknięcia**, jednak należy w nim podać wszystkie dane potrzebne do prawidłowego wysłania pakietu **pukającego**, tak jak to było w przypadku **pukania** z linii komend systemu Linux. Można trzymać obydwa pliki w jednym katalogu, jednak w znacznym stopniu obniża to bezpieczeństwo serwera, gdyż taki plik można uruchomić z każdej maszyny z systemem Windows. Wystarczy tylko, żeby napastnik poznał hasło SSH na naszym serwerze i po **puknięciu** połączył się poprzez np. program putty.

Listing 6. Plik zawierający dane autoryzujące użytkownika

```
grp tomek
port 1001
secret niemamoc
run "ssh %h%"
```

Listing 7. Konfiguracja pliku-klucza

```
#! /bin/sh

knock <ip_serwera> 22
```

```
# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  [redacted]           [redacted]  tcp dpt:ssh
ACCEPT     all  --  anywhere             anywhere            state RELATED,ESTABLISHED

Chain FORWARD (policy DROP)

                                                    state RELATED,ESTABLISHED

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Rysunek 2. Zmiana zasad zapory sieciowej

Zwiększenie bezpieczeństwa

W przypadku takiej potrzeby można jeszcze bardziej podwyższyć poziom zabezpieczeń wdrożonego systemu **port knockingu**. Jednym ze sposobów jest zmiana sposobu uwierzytelniania SSH z metody wykorzystującej hasła na używającą kluczy publicznych i prywatnych, co w znacznym stopniu zwiększy bezpieczeństwo serwera. W takiej sytuacji na serwer zaloguje się jedynie użytkownik, który posiada klucz autoryzujący niezbędny do prawidłowego **puknięcia** i otwarcia portu oraz klucz prywatny, który będzie potrzebny do zalogowania się na usługę SSH. Do wygenerowania takiej pary kluczy wykorzystamy podstawowy program wchodzący w skład pakietu OpenSSH – ssh-keygen. Aby wygenerować parę kluczy (publiczny i prywatny), wydajemy polecenie:

```
# ssh-keygen -t rsa
```

Polecenie `ssh-keygen` wygeneruje dwa pliki. Pierwszy plik **id_rsa** zawiera klucz prywatny (tajny), który nie może być nikomu udostępniany. Drugi **id_rsa.pub** przechowuje klucz publiczny – musimy wysłać go na serwer, z którym zamierzamy ustanawiać nasze połączenia. Kopiujemy klucz publiczny za pomocą polecenia `scp`, także należącego do pakietu OpenSSH:

```
# scp /root/.ssh/id_rsa.pub
                    user@serwer:~/
                    id_rsa.pub
```

Następnie logujemy się na serwer i zakładamy katalog, który będzie przechowywał klucze publiczne autoryzowanych użytkowników, nie zapominając o ustawieniu odpowiednich praw do tego katalogu (chown 700):

```
# mkdir /root/.ssh
```

Następnie dodajemy wygenerowany klucz do listy autoryzowanych kluczy na danym serwerze:

```
serwer:~# cat ~/id_rsa.pub &gt;&gt; ~/.ssh/
                    authorized_keys
```

Inną ciekawą sztuczką jest zmiana standardowego portu usługi SSH (domyślnie port 22) na inny, w żadnym stopniu niesugerujący włamywaczowi, z jaką usługą ma do czynienia. Można także pomyśleć o trudnym do złamania zaszyfrowaniu naszych kluczy na przykład za pomocą programu TruceCrypt, który jest darmowy, a co najważniejsze – jego użycie jest możliwe na różnych systemach operacyjnych. Dzięki takim zabiegom bezpieczeństwo serwera może wielokrotnie się zwiększyć.

Podsumowanie

Stosowanie techniki **port knockingu** jest dobrym i skutecznym rozwiązaniem w zakresie ochrony systemów informatycznych. Dzięki tej metodzie farmy serwerów mogą stać się bezpieczniejsze, a przedstawione w powyższym artykule rozwiązanie to tylko jedna z wielu możliwości konfiguracji, jakie można zastosować. Osoby zainteresowane opisaną metodą zabezpieczeń mogą tworzyć i konfigurować rozwiązania ochronne oparte na **port knockingu** według własnych pomysłów i koncepcji. Oczywiście, jak każde zabezpieczenie, nie jest to bariera nie do przejścia. Zawsze może znaleźć się ktoś, kto do skutku będzie próbował **zbadać bezpieczeństwo** serwera firmowego. Pakiet **puknięcia** musi przejść pewną drogę, zanim dojdzie do celu. Zdecydowanie lepszym rozwiązaniem będzie jednak tradycyjny atak na serwer niż podsłuchanie transmisji **puknięcia** i późniejsze jej rozszyfrowywanie. Nie należy jednak zapominać, że nawet najlepiej skonfigurowana zapora nie jest dobra, jeśli nie współgra z innymi zabezpieczeniami. Uzupełnienie pierścienia zabezpieczeń metodą **port knockingu** buduje barierę trudną do pokonania.

Daniel Suchocki
Daniel Suchocki – z wykształcenia informatyk, jego zainteresowania są ściśle powiązane z zagadnieniem bezpieczeństwa systemów operacyjnych (w szczególności z rodziny Linux). Potrafi o tym opowiadać godzinami, więc nie warto z nim zaczynać rozmowy, jeśli nie ma się wolnego popołudnia. Pracuje jako specjalista w laboratorium informatyki śledczej Mediarecovery.
Kontakt z autorem: dsuchocki@mediarecovery.pl

W Sieci

- http://www.portknocking.org – strona poświęcona technice port knockingu,
- http://doorman.sourceforge.net – strona autora programu Doorman,
- http://iptables.ovh.org – podstawowe reguły iptables,
- http://wendlandt.pl/port-knocking – opis port knockingu na podstawie programu knockd,
- http://mediarecovery.pl – lider informatyki śledczej w Polsce.