{ Marek Serafin }

Bezpieczeństwo sieci firmowej

Kontrola ruchu wychodzącego

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/besifi
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-8322-201-1

Printed in Poland.

Spis treści

ROZDZIAŁ 1.

Wprowadzenie

Niniejsza książka w dużej części porusza temat kontroli ruchu wychodzącego komputerów w firmowej sieci LAN. Blokada — lub przynajmniej kontrola nad tym ruchem — jest warunkiem efektywnego zabezpieczenia sieci firmowej przed niepowołanym dostępem z zewnątrz, a także przed nieautoryzowanym pobieraniem plików i innej szkodliwej treści z sieci. W książce przedstawiam kilka koncepcji filtracji połączeń wychodzących, dzięki którym odzyskasz kontrolę nad poczynaniami użytkowników i zabezpieczysz sieć przed niepowołanym dostępem z zewnątrz.

Motywacją do napisania książki jest doświadczenie, jakie nabrałem podczas wdrażania polityki zwanej „blokuj wszystko". Zmiana podejścia odnośnie do ruchu wychodzącego, a zwłaszcza jego zupełna blokada, nie jest łatwą przeprawą. Na pewno nie obejdzie się bez lamentu użytkowników i dogłębnej analizy połączeń. Dobrze jest rozłożyć temat na raty, aby nie przerwać ciągłości działania firmy. Na pewno prościej będzie osiągnąć cel, wykorzystując firewalle warstwy aplikacyjnej, które zawierają w sobie setki gotowych definicji i sygnatur.

W pierwszych kilku rozdziałach przedstawiam darmowe rozwiązania typu *open source*, gdzie — jak to często w *open source* bywa — trzeba się trochę nagimnastykować, aby osiągnąć cel. Z drugiej strony zyskujemy cenną wiedzę i doświadczenie, które prędzej czy później się przydaje, także podczas debugowania komercyjnych produktów. W rozdziale 7. przedstawiam opis konkretnego wdrożenia dystrybucji OPNsense jako routera i firewalla warstwy aplikacyjnej. Rozdział 8. z kolei zawiera obszerny opis komercyjnego firewalla klasy UTM, za pomocą którego jesteśmy w stanie wdrożyć rozsądną politykę bezpieczeństwa. Rozdział 9. to opis automatyzacji importu certyfikatów SSL i konfiguracji przeglądarek na komputerach użytkowników. W rozdziale 10. zamieszczam szereg uwag i porad ogólnych dotyczących bezpieczeństwa sieci LAN, zdalnego dostępu przez VPN, zabezpieczania serwera Windows. W dodatku na końcu książki zamieszczam najczęściej używane polecenia programu OpenSSL, a także filtry z przykładami do programów Wireshark oraz tcpdump.

Książek traktujących o zabezpieczeniach sieci, protokołach czy firewallach powstało już całe mnóstwo — w tym pokaźne biblie. Dlatego też nie zamieszczam tutaj opisu modelu OSI, podstaw działania protokołów jak ARP, TCP/IP albo działania routingu. Zakładam, że czytelnik jest adminem, czytał o tym setki razy i w większości są to dla niego rzeczy znane. A nawet jeśli nie, to wiedza ta jest powszechnie dostępna, więc uznałem, że nie ma sensu jej powielać i sztucznie „pompować" książki.

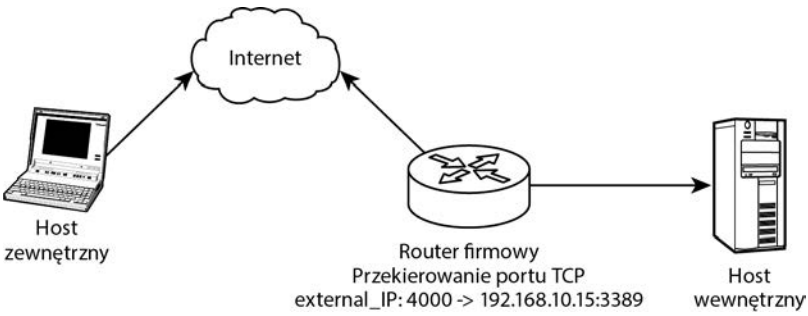
1.1. O co chodzi z tym ruchem wychodzącym?

Typowa sieć LAN odseparowana jest od bezpośredniego połączenia z internetem, tzn. w samej sieci LAN używana jest prywatna adresacja IP — z tzw. puli „nieroutowalnej" (10.x.x.z, 172.16.x.y, 192.168.x.y). W punkcie styku pomiędzy operatorem (ISP) a siecią firmową działa jakiś router (mniejsza o to, czy programowy, czy sprzętowy). Router ten pełni de facto dwie funkcje — umożliwia dostęp do internetu komputerom w sieci LAN (poprzez realizację NAT/PAT) oraz ukrywa je jednocześnie przed bezpośrednim dostępem z całego świata.

Dobrze — skoro komputery w sieci LAN nie są w żaden sposób „osiągalne" (widoczne) bezpośrednio z internetu, co przecież są bezpieczne? Być może kiedyś powyższe stwierdzenie było w pewnym sensie prawdziwe (choć nigdy w 100%), ale już dawno tak nie jest. Postaram się to wytłumaczyć w następnych akapitach.

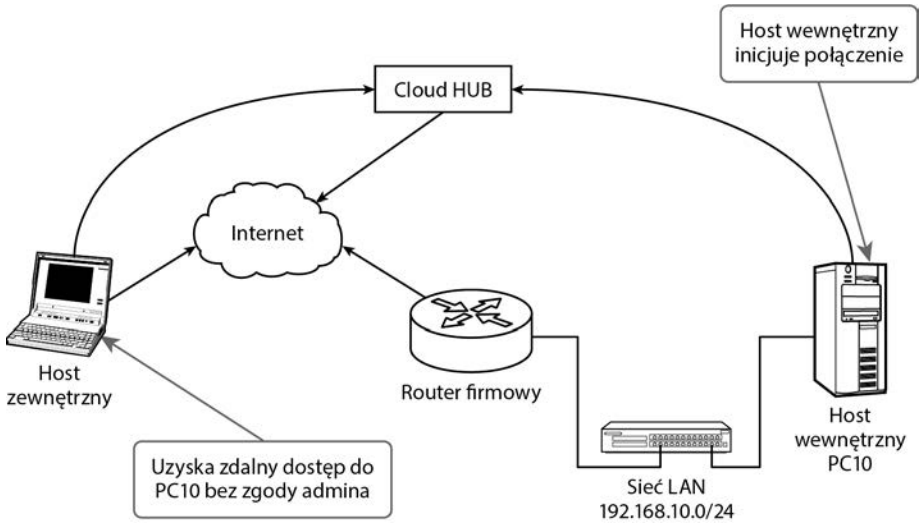
Normalnie aby uzyskać — nazwijmy to — „legalny" zdalny dostęp do komputera znajdującego się w sieci wewnętrznej, należy albo przekierować port na routerze do konkretnej usługi (np. zdalnego pulpitu — port TCP 3389), albo zestawić połączenie VPN z firmą. W obu przypadkach potrzebne są zarówno wola, jak i działanie administratora (patrz rysunek 1.1).

RYSUNEK 1.1. Zdalny dostęp skonfigurowany na żądanie przez administratora sieci



Co się jednak stanie w przypadku, gdy zdalny dostęp zostanie zainicjowany (przez specjalne oprogramowanie) bezpośrednio z komputera w sieci wewnętrznej? Żeby daleko nie szukać podam najpopularniejszy przykład — użytkownik ściąga i uruchamia program typu TeamViewer lub LogeMeIn. Program ten inicjuje połączenie z serwerem TeamViewer i otrzymuje swój unikalny ID, który następnie należy podać w programie klienckim. W momencie podania poprawnego identyfikatora i hasła tworzony jest tunel TCP pomiędzy komputerem klienckim a hostem wewnętrznym — oczywiście bez wiedzy administratora sieci LAN (patrz rysunek 1.2).

Mamy tutaj typowe połączenie tunelowe zestawione bez wiedzy i kontroli administratora. Pal licho, gdy tunel został zestawiony przez pracownika w celu legalnej zdalnej pracy. Znacznie gorsze jest to, że na podobnej zasadzie działają programy szpiegowskie i wirusy umożliwiające przejęcie kontroli nad całym komputerem i siecią LAN. Z tą tylko różnicą, że programy szpiegowskie najczęściej są niewidoczne dla użytkownika zainfekowanego komputera. Włamywacz może zdalnie pracować na komputerze firmowym równolegle z prawowitym użytkownikiem (poprzez wystawioną konsolę/pulpit).



RYSUNEK 1.2. Połączenie tunelowe inicjowane przez hosta wewnętrznego

Drugą istotną różnicą, jaka występuje pomiędzy wirusami a powszechnie znanymi programami typu TeamViewer lub LogMeIn, jest to, że te drugie można w miarę prosto zablokować (choć i z tym bywa różnie, gdy nie ma się odpowiednich narzędzi). Istnieją bowiem sygnatury połączeń, pule adresowe lub wpisy DNS, które należy zablokować, aby uniemożliwić działanie wyżej wspomnianych programów. W przypadku programów typu malware jesteśmy w tej sytuacji zdani tylko na zadziałanie programu antywirusowego. A z nimi bywa różnie.

Zanim przejdę do opisu rozwiązań, jeszcze tytułem wstępu przedstawię kilka bardziej wyrafinowanych możliwości przetunelowania się „na zewnątrz".

1.2. Czym jest tunel lub tunelowanie portów

Dla przypomnienia: tunelowanie to, inaczej mówiąc, opakowanie jednego połączenia (wewnętrznego) drugim (zewnętrznym). Tunelowaniu często towarzyszy szyfrowanie wewnętrznego połączenia, dzięki czemu strony transmisji mają zagwarantowaną poufność przesyłanych danych. Jeśli dodamy do tego np. protokół TLS i certyfikaty SSL, poza poufnością zyskamy także wiarygodność. Dzięki koncepcji infrastruktury klucza publicznego (PKI), czyli certyfikatom, zaufanym urzędom RootCA, mamy pewność, że strona, która przedstawiła się ważnym i podpisanym przez urząd CA certyfikatem, jest rzeczywiście tym, za kogo się podaje. Tunelowanie może odbywać się w warstwie transportowej (4) — mówimy wówczas o tunelowaniu portów (TCP/UDP) lub w warstwie sieci (3). Tunelowanie w warstwie trzeciej (sieci) dokłada dodatkową adresację IP występującą wewnątrz tunelu i tego typu tunel jest de facto połączeniem VPN. Tylko od konkretnej implementacji zależy, jakie algorytmy kryptograficzne zostaną użyte, jaki rodzaj uwierzytelniania stron itp. Jeśli chodzi o tunelowanie portów, to za klasyczny przykład może

posłużyć wszystkim znany protokół HTTPS, którym opakowuje standardowy protokół HTTP, zapewniając poufność (szyfrowanie), jak i wiarygodność (certyfikaty stron). Innym przykładem może być połączenie IMAP na porcie 993, gdzie standardowy protokół IMAP został opakowany protokołem TLS. Albo wynalazek ostatnich lat — protokół *DoH*, czyli tunelowanie zapytań protokołu DNS poprzez HTTPS.

1.3. Dlaczego tunelowanie może być niebezpieczne?

Poza niekwestionowanymi zaletami tunelowania istnieją też zagrożenia wynikające wprost z zasady działania tunelu. Wyobraźmy sobie sieć firmową, w której administrator zablokował prawie wszystkie porty TCP, a dostęp do stron internetowych umożliwił tylko poprzez wewnętrzny serwer proxy (porty 80 i 443 dla połączeń wychodzących są zablokowane dla użytkowników). Podobnie wszystkie porty związane ze zdalnym dostępem zostały zablokowane. Administrator nie zablokował jednak protokołu ICMP, który na ogół służy do diagnostyki połączeń, i uznał go za bezpieczny. W firmie zatrudniono jakiegoś nerda, który postanowił przetunelować się „na świat", aby uzyskać nielimitowany dostęp do stron, a także nieautoryzowany kanał zwrotny do komputera w firmie ☺.

1.4. Tunelowanie TCP po ICMP

Nasz nerd postanowił wykorzystać otwarte pingi i przetunelować się, używając protokołu ICMP (mówiłem, że nerd!). Wykorzystał w tym celu program PingTunnel, który — jak sama nazwa wskazuje — potrafi przetunelować protokół TCP w **ładunku** (ang. *payload*) komunikatów ICMP ☺.

Nasz nerd wystawił na swoim serwerze uruchomiony program PingTunnel nasłuchujący połączeń — patrz listing 1.1.

LISTING 1.1. Uruchamiamy nasłuchiwanie serwera ptunnel

```
root@cd01:/# ptunnel -c eth0 -x mySECRET2022
[inf]: Starting ptunnel v 0.72.
[inf]: (c) 2004-2011 Daniel Stoedle, <daniels@cs.uit.no>
[inf]: Security features by Sebastien Raveau, <sebastien.raveau@epita.fr>
[inf]: Forwarding incoming ping packets over TCP.
[inf]: Initializing pcap.
[inf]: Ping proxy is listening in privileged mode.
```

Natomiast na komputerze w sieci firmowej uruchomił klienta — patrz listing 1.2.

LISTING 1.2. Na komputerze w sieci firmowej uruchamiamy klienta

```
marek@MS:~$ sudo ptunnel -p 188.113.147.110  -lp 8000 -da 127.0.0.1 -dp 22 -x mySECRET2022
[sudo] password for marek:
[inf]: Starting ptunnel v 0.72.
[inf]: (c) 2004-2011 Daniel Stoedle, <daniels@cs.uit.no>
[inf]: Security features by Sebastien Raveau, <sebastien.raveau@epita.fr>
```

```
[inf]: Relaying packets from incoming TCP streams.
[inf]: Incoming connection.
[evt]: No running proxy thread - starting it.
```

Po chwili po stronie konsoli serwera widzimy nadchodzące połączenie.

```
[inf]: Incoming tunnel request from 86.26.238.26.
[inf]: Starting new session to 127.0.0.1:22 with ID 32931
```

Dzięki zestawionemu tunelowi TCP mógł teraz nawiązać sesję SSH ze zdalnym serwerem. W tym celu na komputerze firmowym wywołał SSH z następującymi parametrami: `ssh username@localhost -p 8000` (patrz listing 1.3).

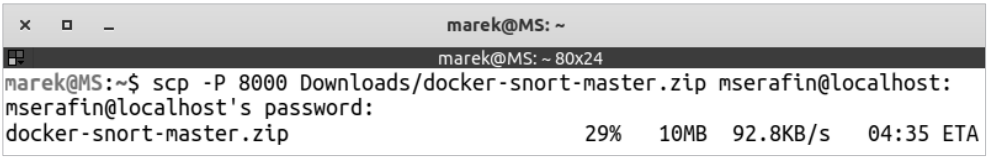
LISTING 1.3. Zestawiona sesja SSH na bazie tunelu TCP over ICMP

```
marek@MS:~$ ssh mserafin@localhost -p 8000
mserafin@localhost's password:
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Dec 21 11:49:59 2022 from 127.0.0.1
```

Jak widać z listingu 1.3, dzięki odblokowanym komunikatom ICMP nerdowi udało się zalogować na prywatny serwer SSH postawiony gdzieś w internecie.

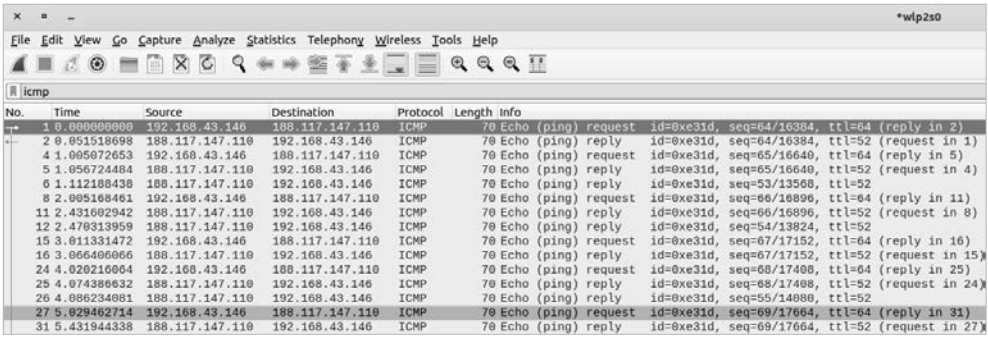
Co prawda transfer pliku przesyłanego przez taki tunel nie powala (patrz rysunek 1.3), ale furtka już jest.



RYSUNEK 1.3. Transfer pliku przez protokół SCP na bazie tunelu TCP over ICMP

Możesz zadać sobie pytanie: no dobra, ma gość SSH do swojego serwera, ale co z tego? Dostęp do SSH daje jednak bardzo wiele możliwości dalszego tunelowania. Polecam zapoznać się z tematyką tunelowania i przekierowania SSH. Użytkownik, mając otwarte połączenie SSH, spokojnie potrafi uzyskać dostęp do dowolnej strony internetowej, tworząc odpowiedni tunel. W tym przypadku ograniczać pewnie będzie go transfer, ale co do zasady — furtka już jest. (W przypadku protokołu ICMP transfer jest ograniczony, z powodu wielkości ładunku, jaki może zostać „przemycony" w pakiecie ICMP).

Na rysunku 1.4 przedstawiono zrzut ekranu analizatora sieci przedstawiający jak połączenie to widziane jest przez administratora sieci. Zgodnie z oczekiwaniami, administrator widzi tylko wymianę pakietów ICMP.

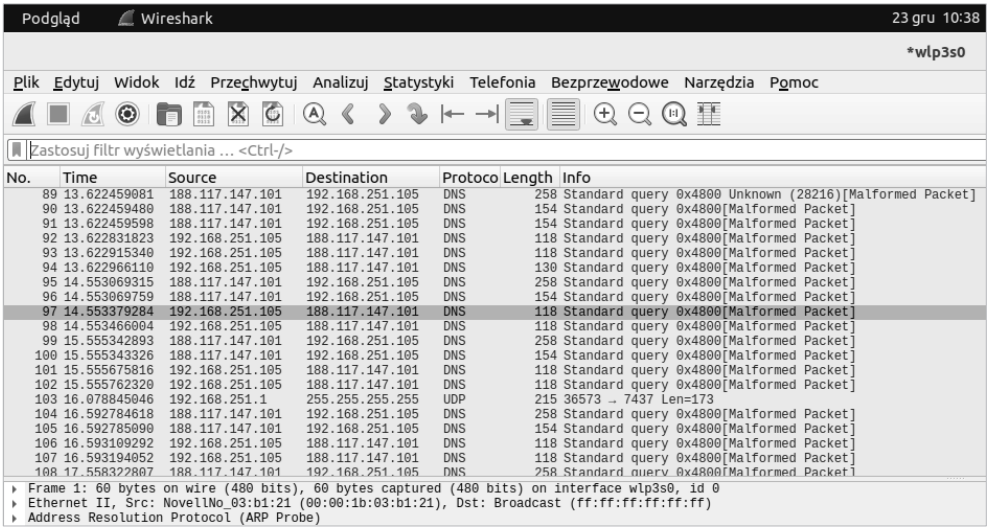


RYSUNEK 1.4. Podgląd tunelu TCP over ICMP w programie WireShark

1.5. Tunelowanie TCP po zapytaniach DNS

Nasz niefrasobliwy pracownik nie był zadowolony z wydajności swojego tunelu. Postanowił więc wykorzystać drugą z furtek, mianowicie odpytania do serwerów DNS.

Teraz ważna uwaga: jeżeli port UDP o nr 53 (zapytania DNS) byłby w firmie otwarty do wszystkich serwerów w internecie, to najprościej byłoby uruchomić serwer OpenVPN-a na porcie 53 i mamy tunel gotowy — żadna większa filozofia. Od strony administratora ruch taki wyglądałby w analizatorze sieci tak, jak pokazano na rysunku 1.5.



RYSUNEK 1.5. Tunel OpenVPN zestawiony na porcie UDP nr 53 — zrzut z analizatora sieci

W tym przypadku pomiędzy komputerem firmowym w sieci LAN a prywatnym serwerem użytkownika w internecie była otwarta komunikacja po porcie 53 UDP (administrator nie zablokował tego portu w ogóle).

Załóżmy teraz gorszy scenariusz. Użytkownicy z sieci firmowej mogą odpytywać tylko wewnętrzne serwery DNS, pozostałe są zablokowane. Okazuje się, że i w takim przypadku istnieje możliwość przetunelowania się. Metoda jest znacznie wolniejsza, bo wykorzystuje tunelowanie w oparciu o zapytania do serwerów DNS, ale przedstawię ją jako ciekawostkę. Wykorzystywana jest tu właściwość samego protokołu DNS, który ma strukturę rozproszoną i hierarchiczną (można wykorzystać serwery pośredniczące do przesłania zapytań).

Aby utworzyć taki tunel, należy po pierwsze posiadać zarejestrowaną jakąś dowolną domenę. Następnie trzeba wydelegować sobie podstrefę DNS w ramach tej domeny, wskazując jej obsługę na adres IP serwera, na którym uruchomiony będzie program do tunelowania. Tworzymy po prostu wpis typu `NS` w edycji naszej domeny — patrz listing 1.4.

LISTING 1.4. Delegacja strefy tunel.mojadomena.pl „w adres IP" zewnętrznego serwera

```
tunel           IN      NS      tunelns.mojadomena.pl.
tunelns         IN      A       188.116.146.101      //<ZEWN_IP_SERWERA>
```

Następnie na wyżej wymienionym serwerze należy zainstalować program iodine (*https://github.com/yarrick/iodine*). Uruchomiony program tworzy pseudoserwer DNS, który służy właśnie do tunelowania ruchu na bazie zapytań (i odpowiedzi) DNS. Właśnie w tym celu należało wydelegować podstrefę obsługiwaną przez ten serwer. Dzięki hierarchicznej strukturze działania protokołu DNS nawet przy braku bezpośredniej komunikacji klienta (komputer wewnątrz firmy) z serwerem możliwe jest zestawienie tunelu (wydelegowana strefa DNS będzie niejako fala nośną dla danych przesyłanych tunelem).

Po stronie serwera należy uruchomić program iodined, przekazując jako parametry klasę adresową wykorzystywaną wewnątrz tunelu, oraz hasło uwierzytelniające (patrz listing 1.5).

LISTING 1.5. Uruchamiamy program po stronie serwera

```
root@cd0:# iodined -f -c -P MyPa$$WORD 172.24.24.1 tunel.mojadomena.pl
Opened dns0
Setting IP of dns0 to 172.24.24.1
Setting MTU of dns0 to 1130
Opened IPv4 UDP socket
Opened IPv6 UDP socket
Listening to dns for domain tunel.mojadomena.pl
```

172.24.24.1 to adres końcówki tunelu po stronie serwera. Można podać dowolną klasę z puli „klas prywatnych" — ważne, aby pula nie kolidowała z adresacją używaną w sieci, która zapewnia dostęp do internetu u klienta (komputer w firmie).

Po stronie klienta (na komputerze w firmie) należy wywołać klienta `iodine`, przekazując jako parametry wywołania ustawione hasło i nazwę domeny — patrz listing 1.6.

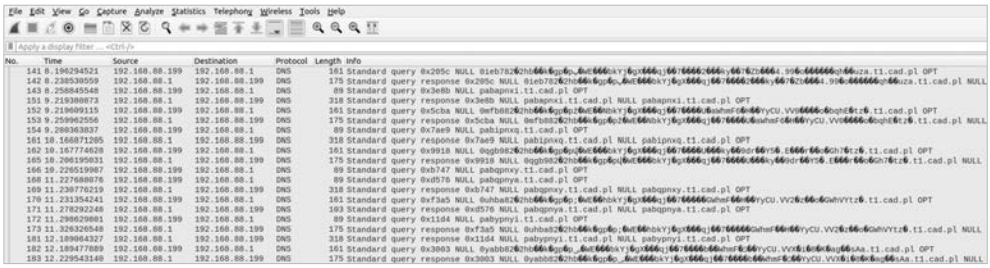
LISTING 1.6. Na komputerze w firmie uruchomiono klienta iodine, przekazując hasło i nazwę domeny

```
marek@MS:~$ sudo iodine -f -P MyPa$$WORD tunel.mojadomena.pl
Opened dns0
Opened IPv4 UDP socket
```

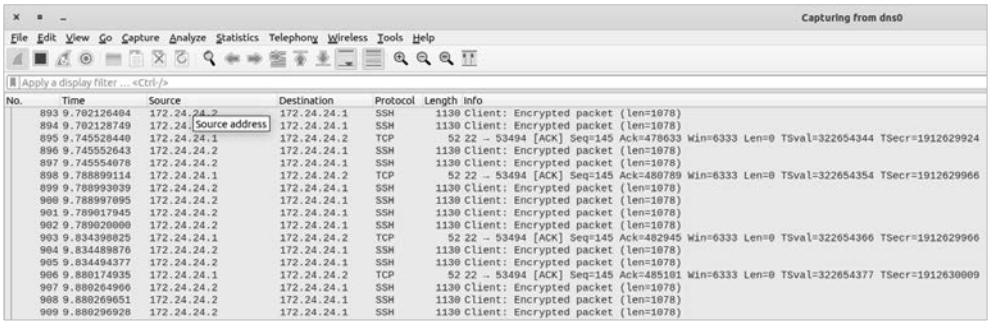
```
Sending DNS queries for tunel.mojadomena.pl to 192.168.88.1
Autodetecting DNS query type (use -T to override).
Using DNS type NULL queries
Version ok, both using protocol v 0x00000502. You are user #1
Setting IP of dns0 to 172.24.24.2
Setting MTU of dns0 to 1130
Server tunnel IP is 172.24.24.1
Testing raw UDP data to the server (skip with -r)
Server is at 188.116.147.110, trying raw login: ....failed
Using EDNS0 extension
Switching upstream to codec Base128
Server switched upstream to codec Base128
No alternative downstream codec available, using default (Raw)
Switching to lazy mode for low-latency
Server switched to lazy mode
Autoprobing max downstream fragment size... (skip with -m fragsize)
768 ok.. 1152 ok.. ...1344 not ok.. ...1248 not ok.. ...1200 not ok.. 1176 ok.. .1188 ok..
↪will use 1188-2=1186
Setting downstream fragment size to max 1186...
Connection setup complete, transmitting data.
```

Po zestawieniu tunelu w systemie pojawi się nowy interfejs *dns0*, który otrzymał adres IP `172.24.24.2`. Adres serwera to `172.24.24.1`. Po tym właśnie adresie odwołujemy się do serwera. Jeśli przyjrzysz się dokładnie listingowi 1.6, to zauważysz, że w pierwszej kolejności klient odpytuje lokalny serwer DNS (tu `192.168.88.1`), aby ustalić adres IP serwera DNS obsługujący strefę tunel.mojadomena.pl. Od swojego serwera DNS otrzymuje adres IP odpowiedzialny za obsługę tej strony (tutaj: `188.116.147.110`). Następnie klient próbuje zestawić bezpośrednie połączenie z serwerem (*Testing raw UDP data to the server*). Niestety połączenia wychodzące do zewnętrznych serwerów DNS są zablokowane, dlatego próba bezpośredniego połączenia klient-serwer kończy się porażką (*Server is at 188.116.147.110, trying raw login: ...failed*). Następnie klient próbkuje wielkość pakietu, jakiego może użyć w komunikacji ze swoim DNS-em i na bazie odpytań DNS zestawia tunel TCP z serwerem. Po to właśnie należało wydelegować strefę DNS w IP serwera, aby pomocnicze serwery (np. firmowe) mogły go odpytywać (a w praktyce przenosić dane). Zauważ, że program wykorzystuje rozszerzenie protokołu EDNS(0). Protokół DNS w swojej pierwotnej postaci zezwalał bowiem na maksymalną wielkość pakietu wynoszącą tylko 512 bajtów. Na rysunku 1.6 przedstawiono widok z analizatora sieci. Zauważ, że administrator widzi tylko komunikację klient (komputer firmowy) → serwer DNS. W analizatorze w ogóle nie widać bezpośredniego połączenia klient → serwer. Dla opisywanego przykładu byłoby to: `192.168.88.199` → `188.116.147.110`, ale takich połączeń nie ma, gdyż są zablokowane (w przeciwieństwie do zrzutu z rysunku 1.5, gdzie pomiędzy klientem a serwerem bezpośrednio zestawiony jest tunel OpenVPN!). Cały ruch odbywa się poprzez zapytania DNS do lokalnego serwera.

Natomiast jeśli w tym samym momencie włączymy nasłuch na interfejsie `dns0` po stronie klienta, zobaczymy właściwe połączenie SSH/SCP — patrz rysunek 1.7. Zwróć też uwagę na adresację IP — wewnątrz tunelu TCP widoczna jest klasa `172.24.24.0/30` użyta na potrzeby tunelu TCP.



RYSUNEK 1.6. Tunel TCP poprzez zapytania DNS. Administrator widzi tylko połączenia z lokalnym serwerem DNS



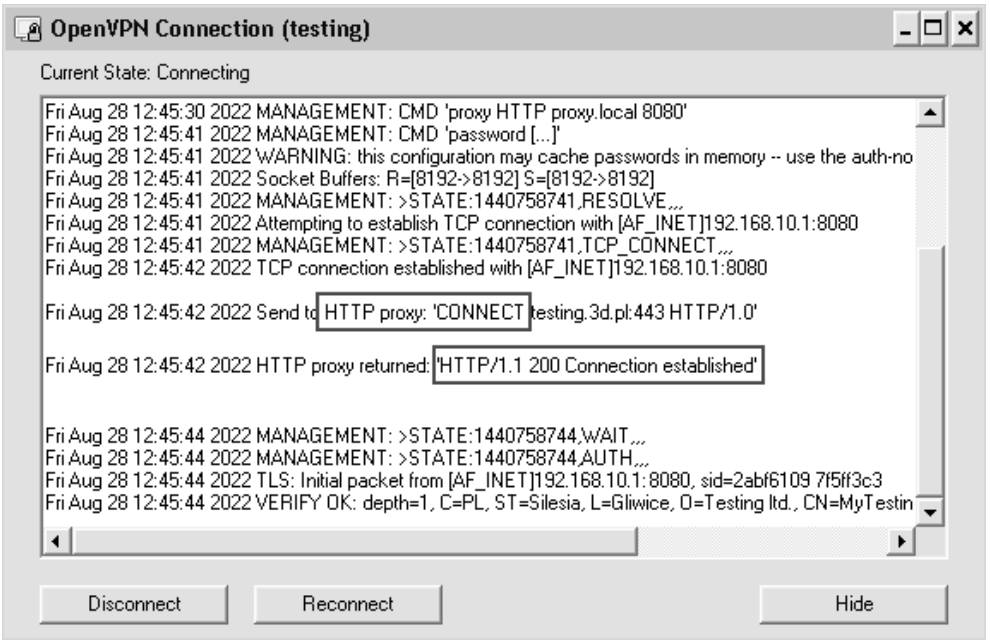
RYSUNEK 1.7. Właściwe połączenie SSH widziane wewnątrz tunelu TCP — zwróć uwagę na adresację IP

Być może przedstawione przeze mnie przykłady są dosyć egzotyczne. Zależało mi tutaj na pokazaniu Ci, jakie są techniczne możliwości wykorzystania otwartych furtek. Zauważ, że gdyby port 53 UDP był rzeczywiście całkowicie otwarty (co często ma miejsce), zamiast bawić się w tunelowanie DNS jakimś egzotycznym programem, można by uruchomić na tym porcie serwer OpenVPN. Przy czym sam program iodine jest dość sprytny. Jeśli wykryje możliwość bezpośredniej komunikacji pomiędzy klientem a serwerem, to zestawia normalny tunel TCP (wówczas zasada działania tego tunelu jest bardzo zbliżona do OpenVPN-a).

1.6. Tunel OpenVPN przez serwer proxy

Na koniec zamieszczam jeszcze całkiem życiowy przykład. Na rysunku 1.8 widzimy zrzut ekranu połączenia OpenVPN z wykorzystaniem nie dość zabezpieczonego serwera proxy i metody `CONNECT`. W tym przypadku klient (sprytny użytkownik) nie miał bezpośredniej komunikacji z serwerem, ale wykorzystał firmowy serwer proxy, który zezwalał na wykorzystanie metody `CONNET` w połączeniach do obcych, niezaufanych serwerów. Więcej na ten temat znajdziesz w rozdziale poświęconemu programowi Squid (rozdział 3.).

Wszystkie z wyżej wymienionych tuneli TCP były utworzone w i tak już względnie zabezpieczonej sieci, ponieważ połączenia wychodzące były poblokowane (należało użyć serwera proxy, protokołów DNS lub ICMP). A co w przypadku sieci, gdzie ma żadnych



RYSUNEK 1.8. Klient OpenVPN w trakcie zestawiania tunelu z wykorzystaniem serwera proxy i metody CONNECT

obostrzeń odnośnie do połączeń wychodzących? W takim przypadku dostęp z zewnątrz jest banalnie prosty. Przetunelować się do takiej sieci można na setki sposobów, i to bez żadnej specjalistycznej wiedzy.

1.7. Co robić, jak żyć?

Jak widzisz już z tego wstępnego rozdziału, aby zabezpieczyć sieć firmową przed nieautoryzowanym dostępem z zewnątrz, trzeba bardzo dokładnie przyjrzeć się połączeniom wychodzącym inicjowanym przez komputery w sieci LAN, a najlepiej w dużej mierze je zablokować.

Z roku na rok programy typu malware stają się coraz bardziej wyrafinowane i często są dostosowywane pod konkretne środowisko (np. Europę Środkowo-Wschodnią) czy wręcz konkretny program lub instytucje i tylko po spełnieniu określonych warunków się aktywują. To dodatkowo utrudnia wykrywalność przez wiodące programy antywirusowe. Niektóre programy z kolei pobierają „kolejne instrukcje" przez internet, więc same w sobie „na pierwszy rzut oka" nie wyglądają podejrzanie. Bywa, że wykorzystują do tego nawet systemy DNS (rekordy TXT zawierające instrukcje poleceń), aby jeszcze bardziej zagmatwać zasadę działania. Z pewnością z każdym rokiem liczba kradzieży, włamań i zagrożeń będzie rosła, gdyż z jednej strony stoją za tym zorganizowane grupy przestępcze zatrudniające najlepszych specjalistów, a z drugiej bardzo poważne pieniądze pozyskiwane z okupów od zhakowanych firm. Sprawy nie ułatwia też (po)pandemiczna

rzeczywistość, gdzie bardzo wielu użytkowników łączy się z firmą z domu przez VPN, a na temat ich sieci domowej niewiele wiemy.

Warto zatem podjąć konkretne kroki, aby zmniejszyć ryzyko ataku i zyskać kontrolę nad tym, z czym łączą się komputery firmowe.

Wdrożenie restrykcyjnej polityki odnośnie do połączeń wychodzących nie będzie łatwą przeprawą. Zwłaszcza jeśli do tej pory użytkownicy mieli wszystko odblokowane. Warto jednak podjąć trud, gdyż w nagrodę zyskasz kontrolę nad przepływem pakietów w sieci, którą administrujesz.

W dalszej części książki opiszę rożne podejścia do tematu blokady połączeń. Zacznę od opisu narzędzi *open source* i pełnej blokady połączeń, by dojść do opisu firewalli warstwy aplikacyjnej, który na podstawie sygnatur połączeń potrafi odróżnić i sklasyfikować połączenia, które dla tradycyjnego firewalla wyglądają tak samo.

Podpowiem, na co zwrócić uwagę, jak zrobić to w miarę bezboleśnie, jednocześnie nie paraliżując działania firmy ☺.

Jako gratis na FTP-ie dołączam listę adresów URL, które stosuję w wyjątkach (tzw. biała lista), chodzi o adresy banków, instytucji rządowych, finansowych, głównych systemów płatności i sklepów internetowych. Lista ta oczywiście dostosowana jest do polskich realiów i zawiera adresy rodzimych serwisów (choć, rzecz jasna, nie tylko).

Starałem się, aby przykłady były konkretne i życiowe. Mam nadzieję, że wiedzę wykorzystasz jak najszybciej w praktyce. Życzę Ci przyjemnej lektury i sieci wolnej od włamów!

Bezpieczeństwo sieci firmowej

w dużym stopniu zależy od kontroli, jaką administrator ma nad połączeniami inicjowanymi przez komputery użytkowników. Jej brak umożliwia użytkownikom otwieranie adresów niebezpiecznych stron, pobieranie zainfekowanych plików, a w konsekwencji naruszenie bezpieczeństwa całej sieci. W dobie zmasowanych ataków typu ransomware wprowadzenie kontroli nad połączeniami wychodzącymi to absolutna konieczność.

Autor książki nakreśla w niej zagrożenia, a także omawia różne koncepcje blokady połączeń i filtracji stron WWW z wykorzystaniem dostępnych na rynku rozwiązań. Przedstawia zarówno darmowe narzędzia open source, na przykład Squid, E2guardian, OPNsense, jak i produkty komercyjne — Fortigate UTM czy Web Safety. To propozycja dla administratorów sieci w małych i średnich firmach, jak również w instytucjach, urzędach, szkołach i na uczelniach. Autor od lat zajmuje się administrowaniem sieciami i systemami komputerowymi, jego wcześniejsza pozycja, *Sieci VPN. Zdalna praca i bezpieczeństwo danych*, uzyskała status bestsellera.

Dzięki książce poznasz:

- **najlepsze praktyki zabezpieczania sieci**
- **różne koncepcje filtrowania ruchu**
- **metody blokowania niepożądanych połączeń**
- **metody ochrony użytkowników przed niepożądaną treścią**

Marek Serafin jest doświadczonym administratorem sieci i systemów komputerowych. Linux, serwerownia i konfiguracja switchów to jego naturalne środowisko. Autor bestsellera *Sieci VPN. Zdalna praca i bezpieczeństwo danych*. W czasie wolnym lutuje układy elektroniczne i programuje mikrokontrolery. Jest miłośnikiem kotów.

Helion

helion.pl

HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-8322-201-1

9 788383 222011

Cena: 69,00 zł