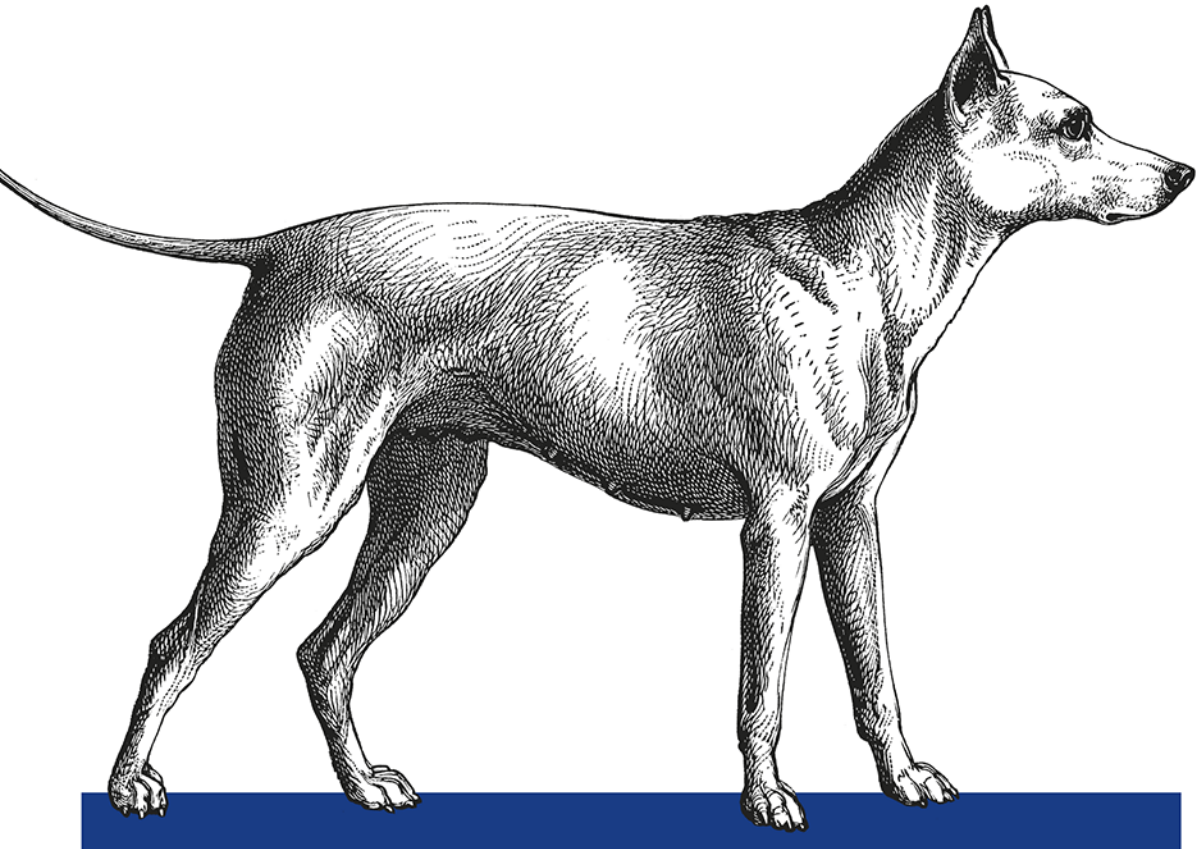
O'REILLY®



Kali Linux

TESTY BEZPIECZEŃSTWA, TESTY PENETRACYJNE
I ETYCZNE HAKOWANIE

Helion

Ric Messier

Tytuł oryginału: Learning Kali Linux: Security Testing, Penetration Testing, and Ethical Hacking

Tłumaczenie: Andrzej Watrak

ISBN: 978-83-283-5426-5

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

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

Printed in Poland.

Spis treści

ROZDZIAŁ 2.

Podstawy testowania bezpieczeństwa sieci

Testy bezpieczeństwa to szeroki temat obejmujący wiele różnych zagadnień. Niektóre z nich dotyczą bezpieczeństwa sieci, ale celem testów nie zawsze jest sprawdzanie podatności systemów na włamania. Testy mogą dotyczyć możliwości pogorszenia jakości usług sieciowych, na przykład przerwania działania usługi lub utraty jej dostępności. Brak usługi oznacza zagrożenie bezpieczeństwa systemu. Dlatego test obciążeniowy usługi jest ważnym elementem testu bezpieczeństwa.

Aby przeprowadzić test bezpieczeństwa obejmujący nie tylko aplikacje, lecz także różne elementy sieciowe, trzeba znać strukturę stosu protokołów. Jedną z definicji stosu protokołów sieciowych, a dokładniej interakcji między nimi, jest model OSI (ang. *Open Systems Interconnection*, łączenie systemów otwartych). W tym modelu komunikacja sieciowa jest podzielona na osobne bloki funkcjonalne, w których do tworzonych pakietów danych dodawane są różnego rodzaju informacje. Ponadto model opisuje interakcje pomiędzy systemem operacyjnym a powyższymi blokami funkcjonalnymi.

Podczas wykonywania testów obciążeniowych nie tylko uzyskuje się mnóstwo informacji o faktycznej wydajności systemu i aplikacji, lecz także otrzymuje się nieoczekiwane wyniki. Testy mogą polegać na zamierzonym łamaniu zasad komunikacji klientów z aplikacją lub systemem. Wiele ataków przeprowadzanych jest właśnie w ten sposób. W wyniku testów aplikacje mogą ulegać awariom, na przykład przerywać swoje działanie lub zgłaszać wyjątki, które potencjalnie mogą zostać wykorzystane do włamania się do aplikacji lub systemu.

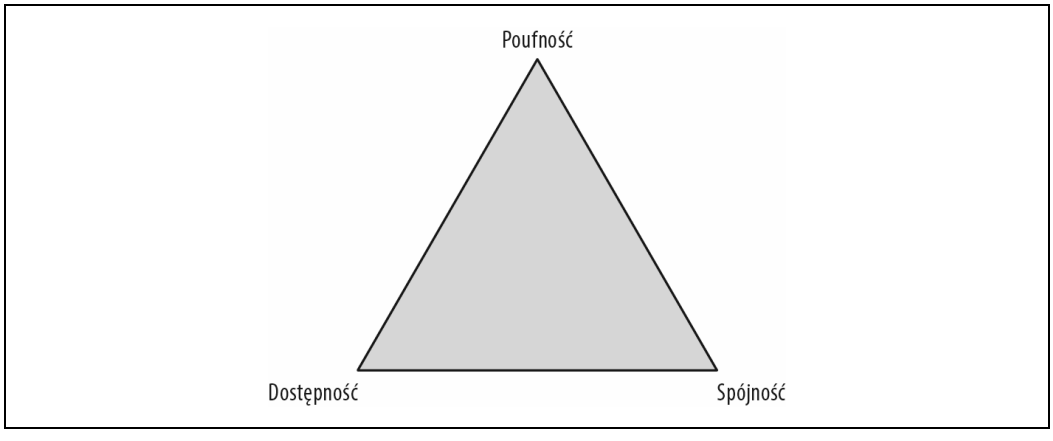
Testy bezpieczeństwa

Wiele osób, używając terminu *testy bezpieczeństwa*, ma na myśli testy penetracyjne, których celem jest uzyskanie dostępu do systemu z najwyższymi możliwymi uprawnieniami. Jednak testy bezpieczeństwa obejmują nie tylko „otwieranie pudełek". Systemy i oprogramowanie są zabezpieczane na wiele innych sposobów, których nie sprawdzają testy penetracyjne. Zanim zajmiemy się możliwościami testowania bezpieczeństwa sieci za pomocą narzędzi dostępnych w systemie Kali, musimy dokładnie określić, czym jest bezpieczeństwo i jego testy w kontekście tego systemu.

Specjaliści od IT, w szczególności w instytucjach wydających certyfikaty bezpieczeństwa, często odnoszą się do tzw. **triady**. Niektórzy dodają do niej dodatkowe elementy, ale rdzeń bezpieczeństwa informacji zawsze pozostaje ten sam: poufność, spójność i dostępność. Jeżeli zostanie naruszona którakolwiek z tych cech, zagrożone będzie bezpieczeństwo systemu lub oprogramowania. Testy

bezpieczeństwa powinny uwzględniać te trzy cechy i nie ograniczać się jedynie do pozyskiwania informacji typowych dla testów penetracyjnych.

Jak wiesz, triadę zazwyczaj przedstawia się jako trójkąt równoboczny. Oznacza to, że wszystkie trzy elementy są tak samo ważne. Co więcej, jeżeli któregokolwiek z tych elementów zabraknie, trójkąt przestanie istnieć. Rysunek 2.1 w sposób typowy przedstawia triadę jako trójkąt o równych bokach. Każda z podanych cech ma równie istotny wpływ na rzetelność i wiarygodność informacji. Dzisiaj, gdy dane biznesowe i osobowe są przechowywane w formie cyfrowej, bardzo ważne jest, aby były one dostępne, poufne (gdy jest taka potrzeba) oraz spójne.



Rysunek 2.1. Triada poufność — spójność — dostępność

Większość firm przetwarza własne poufne dane. Poufne są też dane osobowe, na przykład numery dowodów osobistych, hasła, numery PESEL, dane medyczne i różnego innego rodzaju informacje. Firmy muszą chronić swoją własność intelektualną. Mają swoje sekrety handlowe, których upublicznienie negatywnie wpłynęłoby na ich działalność. Ograniczona dostępność tych informacji, niezależnie od ich charakteru, jest określana mianem **poufności**. Jeżeli informacja wydostanie się poza miejsce, w którym jest bezpiecznie przechowywana, zostanie naruszona jej poufność. Poufność to podstawowy element bezpieczeństwa informacji naruszany w przypadku jej kradzieży, co się niezwykle często zdarza w różnych firmach i instytucjach; przykładem jest Target, Equifax, Sony i amerykański Office of Personnel Management (Urząd Zarządzania Kadrami). Jeżeli dane osobowe zostaną wykradzione, zostanie naruszone ich bezpieczeństwo.

Przechowując jakąś informację, zazwyczaj musimy jednocześnie mieć możliwość jej odczytywania. Dane mogą zostać uszkodzone lub zmienione z różnych powodów, nie zawsze wynikających z czyjejś złej woli. Zagrożenie bezpieczeństwa nie zawsze jest związane z wrogimi intencjami. Oczywiście wspomniana wcześniej kradzież informacji stanowiła zagrożenie. Jednak niewłaściwie funkcjonująca pamięć komputera może spowodować uszkodzenie danych zapisanych na dysku. Wiem to z własnego doświadczenia. Awaria dysku lub innego nośnika również może być przyczyną uszkodzenia danych. Oczywiście w szczególnych przypadkach rozmyślnie szkodliwe operacje mogą skutkować uszkodzeniem i zakłóceniem przetwarzania danych. Niezależnie jednak od tego, co było przyczyną, efektem jest zakłócenie ich spójności. **Spójność** oznacza, że wszystkie informacje znajdują się w stanie, w którym powinny się znajdować.

Rozważmy jeszcze **dostępność** danych. Jeżeli komputer spadnie na podłogę i wypadnie wtyczka z gniazdka zasilającego, komputer przestanie być dostępny (mam na myśli komputer stacjonarny, bez baterii). Podobnie, jeżeli zatrzask w przewodzie sieciowym ulegnie uszkodzeniu i przewód wypadnie z gniazdka w ścianie lub z karty komputera, Twój system przestanie być dostępny w sieci. Oczywiście zdarzenie to będzie miało wpływ przede wszystkim na Twoją pracę, ale może również dotyczyć innych osób, które korzystają z danych zapisanych na Twoim komputerze. Jeśli serwer ulegnie awarii, zakłócona zostanie dostępność znajdujących się na nim informacji. Jeżeli haker spowoduje awarię usługi lub całego systemu, nawet krótkotrwałą, zakłócona zostanie dostępność danych, co może mieć poważny wpływ na funkcjonowanie firmy. Klienci nie będą wtedy mieć dostępu do usług i trzeba będzie zainwestować w ludzi i zasoby, aby usługi znów zaczęły działać i były dostępne. Tak jest w przypadku banków, które stały się obiektami zmasowanych, długotrwałych ataków DoS (ang. *denial of service*, blokada usługi). Ataki wprawdzie mogą się nie powieść, ale na pewno odpieranie ich ma negatywny wpływ na funkcjonowanie firmy.

Testowanie bezpieczeństwa informacji oznacza testowanie jej opisanych wyżej cech. Forma testów nie jest istotna. Testy bezpieczeństwa sieci mogą dotyczyć podatności usługi na ataki, skuteczności szyfrowania informacji i innych czynników. Na początku do przeprowadzenia testów bezpieczeństwa sieci potrzebne są narzędzia do wykonywania testów obciążeniowych. Pożądane są też narzędzia, które mogą spowodować awarię sieci. Choć na przestrzeni lat wiele błędów w stosie protokołów sieciowych i systemach operacyjnych zostało usuniętych, wciąż mogą być stosowane proste urządzenia, które są bardziej podatne na ataki sieciowe. Takimi urządzeniami są na przykład drukarki, telefony IP, termostaty, lodówki, a także każdy inny sprzęt podłączany do internetu.

Testy bezpieczeństwa sieci

Z siecią jesteśmy związani na śmierć i życie. Mnóstwo danych osobowych jest dzisiaj gromadzonych w internecie lub jest dostępnych przez internet w tzw. **chmurze**. Dzisiaj oczekujemy, że wszystko będzie można znaleźć w sieci. Dlatego bardzo ważne jest, aby urządzenia, z których korzystamy, były odporne na ataki.

Monitoring

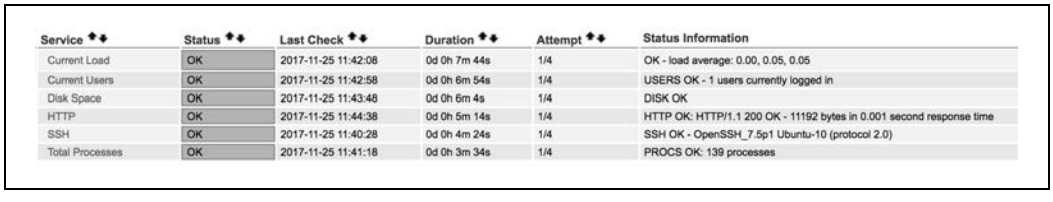
Zanim zaczniemy wykonywać testy, musisz poznać ważny temat monitoringu. Przede wszystkim testowanie bezpieczeństwa sieci w Twojej lub klienta firmie nie może skutkować awarią wszystkich urządzeń, chyba że wyraźnie zostaniesz o to poproszony. Niemniej jednak pomimo zachowania najwyższej ostrożności coś może pójść nie tak i usługi albo systemy zostaną zablokowane. Dlatego bardzo ważne jest powiadomienie osób opiekujących się infrastrukturą IT, aby sprawowały kontrolę na systemami i usługami. Właściciele firm nie chcą, aby zakłócana była obsługa ich klientów, dlatego życzą sobie, by podczas testów były obecne osoby, które w razie potrzeby zrestartują usługi.

Niektóre firmy chcą sprawdzać skuteczność pracowników zarządzających siecią. Oczekują wtedy włamania się do jakiegoś systemu lub wywołania jego awarii bez powodowania długotrwałych zakłóceń ani tym bardziej trwałych uszkodzeń. W takim wypadku nie informuj o testach nikogo oprócz menedżerów, którzy zlecili Ci ich wykonanie. W większości przypadków firmy chcą mieć pewność, że ich środowisko produkcyjne działa nieprzerwanie.

Jeżeli w przeprowadzanie testów zaangażowany jest zespół IT, musisz w jakiś sposób monitorować systemy. Możesz w tym celu śledzić wpisy w dziennikach, co jest ogólnie zalecaną metodą. Dzienniki jednak nie są niezawodnym źródłem informacji. Jeśli będziesz w stanie wywołać awarię usługi, może ona nie zdążyć zapisać w dzienniku użytecznej informacji o problemie. Nie oznacza to jednak, że możesz zrezygnować ze śledzenia dzienników. Pamiętaj, że celem testów jest poprawa bezpieczeństwa firmy, w której pracujesz. Dzienniki mogą zawierać kluczowe informacje o tym, co się działo z daną usługą, zanim uległa awarii. Problem może polegać nie na tym, że jej procesy przestały działać, ale że działały niezgodnie z oczekiwaniami. W takich sytuacjach dzienniki odgrywają istotną rolę w diagnozowaniu, co się działo z aplikacją.

Czasami do kontrolowania działania procesów stosowane są programy nadzorujące (ang. *watchdog*). Jeżeli jakiś proces zostanie przerwany, jego identyfikator PID nie zostanie usunięty z listy i program nadzorujący uruchomi go ponownie. Ponadto program nadzorujący można wykorzystywać do sprawdzania, czy proces ulegnie awarii. Nawet jeżeli nie trzeba będzie go ponownie uruchamiać, można kontrolować tablicę procesów i sprawdzać, czy coś niepokojącego się z nimi nie dzieje.

Niekontrolowane procesy mogą zająć wszystkie zasoby systemu. Dlatego bardzo ważne jest obserwowanie obciążenia procesora i zajętości pamięci. Można użyć do tego celu bezpłatnych narzędzi monitorujących, narzędzi komercyjnych lub w przypadku systemów Windows i macOS wbudowanych narzędzi diagnostycznych. Jednym z popularnych programów monitorujących jest **Nagios**. Zainstalowałem go w jednym ze swoich wirtualnych systemów. Rysunek 2.2 przedstawia widok jednego z okien. Bez dodatkowej konfiguracji Nagios monitoruje między innymi procesy, obciążenie procesora oraz stany usług SSH i HTTP.



Service	Status	Last Check	Duration	Attempt	Status Information
Current Load	OK	2017-11-25 11:42:08	0d 0h 7m 44s	1/4	OK - load average: 0.00, 0.05, 0.05
Current Users	OK	2017-11-25 11:42:58	0d 0h 6m 54s	1/4	USERS OK - 1 users currently logged in
Disk Space	OK	2017-11-25 11:43:48	0d 0h 6m 4s	1/4	DISK OK
HTTP	OK	2017-11-25 11:44:38	0d 0h 5m 14s	1/4	HTTP OK: HTTP/1.1 200 OK - 11192 bytes in 0.001 second response time
SSH	OK	2017-11-25 11:40:28	0d 0h 4m 24s	1/4	SSH OK - OpenSSH_7.5p1 Ubuntu-10 (protocol 2.0)
Total Processes	OK	2017-11-25 11:41:18	0d 0h 3m 34s	1/4	PROCS OK: 139 processes

Rysunek 2.2. Monitoring zasobów systemu

Jeżeli z jakiegoś powodu nie będziesz mógł współpracować z zespołem IT, a nie masz bezpośredniego dostępu do testowanych systemów, będziesz musiał mieć możliwość przynajmniej zdalnie kontrolować usługę. Opisywane tu narzędzia do testowania sieci mogą przestać odbierać odpowiedzi z testowanych usług, co może — ale nie musi — być oznaką awarii usługi. Może to być symptomem problemu z samym narzędziem monitorującym albo skutkiem zastosowania mechanizmu blokującego niepożądaną aktywność w sieci. W takich sytuacjach ważne jest, aby dało się bezpośrednio sprawdzić, czy usługa przestała działać.

Notowanie to podstawa

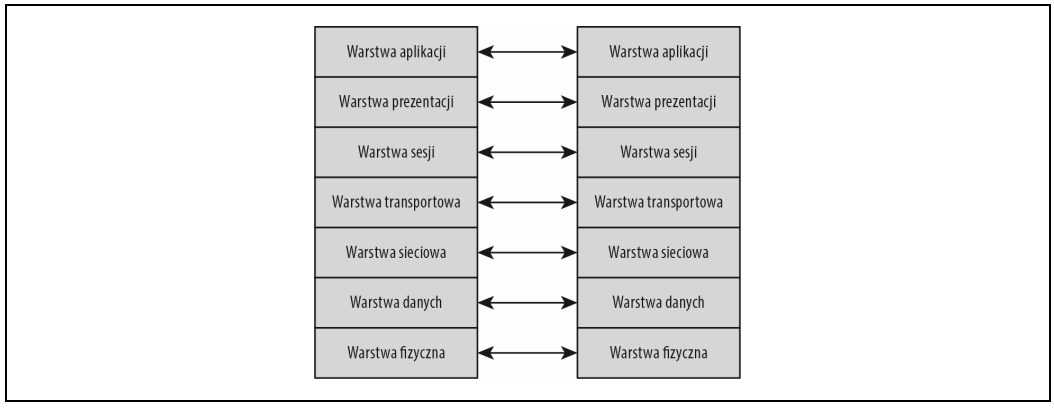
Jeżeli podczas testowania usługi stwierdzisz, że przestała działać, postaraj się w miarę możliwości jak najdokładniej zanotować czas awarii. Informowanie klienta lub pracodawcy o awarii usługi niewiele wnosi, ponieważ mogą oni nie wiedzieć, jak rozwiązać problem. Szczegółowe notatki przydadzą Ci się, gdy będziesz musiał poinformować zespół IT, co dokładnie robiłeś w chwili awarii. Dzięki temu będzie można określić przyczyny problemu i go rozwiązać.

Testy można wykonywać ręcznie za pomocą narzędzi takich jak `netcat`, a nawet `telnet`. Jeżeli uda Ci się nawiązać połączenie z wykorzystywanym przez usługę portem, będziesz miał pewność, że usługa reaguje. Taka ręczna weryfikacja (szczególnie gdy jest wykonywana za pomocą osobnego systemu, który na pewno nie jest blokowany) pozwoli wyeliminować fałszywe symptomy problemów. Generalnie testy bezpieczeństwa często polegają na eliminowaniu fałszywych alarmów wynikających ze stosowania różnych narzędzi. Monitorując i weryfikując wyniki, możesz mieć pewność, że dane, które później przedstawisz pracodawcy lub klientowi, będą wiarygodne i pozwolą mu podejmować odpowiednie decyzje.

Warstwy

Jak stwierdził Osioł w firmie *Shrek*: warstwy są bardzo ważne. W rzeczywistości Shrek mówił, że ogry mają warstwy, a Osioł mówił, że mają je torty. Shrek porównywał ogra do cebuli, a Osioł uważał, że wszyscy lubią torty, bo są lepsze od cebuli. Do dzisiaj słyszę głos Osła. Oczywiście ani ogry, ani cebule nie są tutaj ważne, ale torty owszem. Mówiąc o sieciach i komunikacji pomiędzy systemami, zazwyczaj mamy na myśli warstwy. Gdy wyobrazisz sobie siedmiowarstwowy tort, łatwiej Ci będzie zrozumieć działanie sieci. Dodatkowo wyobraź sobie dwa kawałki takiego tortu. W końcu dwa kawałki są lepsze niż jeden, prawda?

Rysunek 2.3 jest prostą ilustracją siedmiowarstwowego modelu OSI i komunikacji pomiędzy poszczególnymi warstwami dwóch osobnych systemów. Aby rzecz wyglądała ciekawiej, możesz sobie wyobrazić, że pomiędzy warstwami znajduje się dżem, który je ze sobą skleja. Komunikujące się ze sobą warstwy w obu systemach są dokładnie takie same. Gdy jeden kawałek tortu wysyła komunikat do drugiego, wykorzystuje do tego celu dwie odpowiadające sobie warstwy.



Rysunek 2.3. Model OSI i komunikacja między systemami

Kontynuujmy przykład z tortami. Na samym dole znajduje się *warstwa fizyczna*, którą można porównać do spodu tortu. Jest to miejsce, w którym system łączy się z siecią, w naszym słodkim przykładzie jest to taca, na której leży tort. Podobnie jak w przypadku tortu pomiędzy warstwą fizyczną systemu a siecią niczego nie ma. Za pomocą przewodu łączy się interfejs sieciowy systemu z gniazdkiem w ścianie. To jest wszystko, co można zrobić w warstwie fizycznej. W przypadku tortu jego spód spoczywa bezpośrednio na tacy i niczego pomiędzy nimi nie ma.

Następną warstwą jest krem, czyli *warstwa danych*. Od warstwy fizycznej oddziela ją dżem, dzięki któremu system operacyjny może obie warstwy rozróżnić. W warstwie danych stosowane są adresy MAC (ang. *Media Access Control* — **sterowanie dostępem do medium**). Pierwsze trzy bajty adresu to identyfikator producenta, tzw. OUI (ang. *Organizationally Unique Identifier* — **unikatowy identyfikator organizacji**). Pozostałe trzy bajty (cały adres MAC składa się z sześciu bajtów) tworzą unikatowy identyfikator interfejsu sieciowego. Obie części razem wzięte tworzą adres MAC. Wszelka komunikacja w obrębie lokalnej sieci odbywa się w tej warstwie. Jeżeli warstwa kremu w jednym kawałku tortu chce porozmawiać z warstwą kremu w drugim kawałku (kto inny jak nie krem zrozumie inny krem?), musi użyć adresu MAC, ponieważ jest to jedyny adres, który rozumieją interfejsy sieciowe dwóch systemów. Adres jest przypisany do interfejsu, dlatego jest nazywany adresem fizycznym. Listing 2.1 przedstawia wynik użycia polecenia `ifconfig`. Adres MAC jest drugim ciągiem od lewej strony.

Listing 2.1. Adres MAC

```
ether 52:54:00:11:73:65  txqueuelen 1000  (Ethernet)
```

Następną warstwą, do której trzeba się dostać poprzez dżem, jest biszkopt, czyli *warstwa sieciowa*. W tej warstwie stosowane są adresy IP, za pomocą których dane można wysyłać poza lokalną sieć. Adresy MAC w odróżnieniu od adresów IP nigdy nie wydostają się poza lokalną sieć. Torty danej cukierni sprzedawane w różnych punktach są dokładnie takie same, więc mogą się ze sobą komunikować za pomocą tej warstwy i adresów IP. Listing 2.2 przedstawia taki przykładowy adres. Składa się on z 4 bajtów, zwanych również **oktetami**, ponieważ każdy zbudowany jest z ośmiu bitów. W poniższym przykładzie jest to adres IP w wersji 4 (ciąg drugi od lewej). Adres w wersji 6 składa się z 16 bajtów (128 bitów). Podobnie jak poprzednio listing przedstawia wynik użycia polecenia `ifconfig`.

Listing 2.2. Adres IP

```
inet 192.168.86.35  netmask 255.255.255.0  broadcast 192.168.86.255
```

Przechodzimy do następnej warstwy, którą jest galaretka (*warstwa transportowa*). Tak, to jest bardzo nietypowy tort, ale niech tak już zostanie. W tej warstwie stosowane są porty, czyli jeszcze innego rodzaju adresy. Możesz je sobie wyobrazić w następujący sposób. Gdy wejdziesz do cukierni, musisz odnaleźć odpowiednią półkę. Tak samo jest z portami. Jeżeli znajdziesz cukiernię o żądanym adresie IP, musisz w niej znaleźć odpowiednią półkę, czyli port. Port łączy działającą usługę (program) przypisaną do danej półki. Niektóre usługi wykorzystują tzw. dobrze znane porty, czyli przeznaczone do określonych zastosowań. Usługa (na przykład strona WWW) może wykorzystywać dowolny port, ale najczęściej jest to port dobrze znany, ponieważ wiadomo wtedy, co usługa oferuje.

Warstwa piąta jest dość trudna, ponieważ zazwyczaj jest niewłaściwie opisywana. Jest to mus owocowy, bo w torcie musi być trochę owoców lub choćby ich smaku. Jest to *warstwa sesji* odpowiedzialna za długotrwałą komunikację i synchronizację stron. Wyobraź sobie, że Ty i ja zaczynamy w tej samej chwili jeść swoje kawałki toru (zaczynamy się komunikować). Zadaniem tej warstwy jest dopilnowanie, abyśmy jedli w tym samym tempie i skończyli w tym samym momencie. Jeżeli któryś z nas chce na chwilę przerwać jedzenie i napić się wody, to warstwa sesji wymusza, abyśmy obaj zrobili to samo w tej samej chwili. Jeżeli któryś z nas zechce napić się mleka zamiast wody, warstwa ta będzie nas synchronizować, abyśmy zaczęli i skończyli pić w tym samym czasie i tak samo wyglądali podczas jedzenia. Bo o wygląd tu głównie chodzi.

Przechodzimy do warstwy orzechów, bo czymże byłby tort bez masła orzechowego? To jest *warstwa prezentacji*, która dba o to, aby wszystko wyglądało tak, jak trzeba, na przykład: aby nie było łupinek orzechów i aby to, co wkładamy do ust, wyglądało jak tort.

Ostatnią warstwą jest bita śmietana, czyli *warstwa aplikacji*, znajdująca się najbliżej konsumenta tortu (użytkownika). Warstwa ta odbiera to, co oddaje warstwa prezentacji i przekazuje użytkownikowi do skonsumowania. Ważnym elementem w tej analogii jest widelec, którym odkrawa się kawałki tortu począwszy od warstwy najwyższej, a na najniższej skończywszy. Natomiast do ust wędruje najpierw najniższa warstwa. Tak samo są wysyłane i odbierane komunikaty. Powstają one w warstwie aplikacji, po czym są przesyłane w dół stosu. Komunikat jest *konsumowany* w najniższej, fizycznej warstwie, a podczas przesyłania do wyższych warstw są z niego zdejmowane kolejne nagłówki.

Testowanie sieci polega na pracy z różnymi warstwami tortu. Dlatego ważna jest znajomość każdej z nich. Musisz znać wymagania wszystkich warstw i umieć ocenić, czy działają one prawidłowo. Podczas testów będziesz miał do czynienia z różnymi warstwami, ale zazwyczaj każde narzędzie będzie przeznaczone dla jednej określonej warstwy. Komunikacja sieciowa polega na zjadaniu całego tortu, ale czasami trzeba skupiać się na konkretnej warstwie w oderwaniu od reszty tortu i sprawdzać, czy ma właściwy smak.

Testy obciążeniowe

Niektóre programy, a nawet urządzenia, źle sobie radzą poddane dużym obciążeniom. Dzieje się tak z różnych powodów. Sprzęt o konkretnym przeznaczeniu lub urządzenia IoT (ang. *Internet of Things*, internet rzeczy) z wielu różnych przyczyn mogą nie być w stanie obsłużyć dużego ruchu. Na przykład układy elektroniczne w interfejsie sieciowym mogą nie być wystarczająco wydajne, ponieważ producent zakładał, że nigdy nie będą obsługiwać dużego ruchu. Programy mogą zwierać błędy. Nawet oprogramowanie wbudowane w sprzęt może przysparzać problemów, jeżeli będzie niewłaściwie zaprojektowane. Dlatego ważne jest, aby testerzy bezpieczeństwa mieli pewność, że systemy, za które są odpowiedzialni, nie ulegną awarii, gdy stanie się coś złego.

Test bezpieczeństwa można wprost porównać do ataku polegającego na zalaniu urządzenia pakietami. Jednak są również inne sposoby obciążania aplikacji. Jeden z nich polega na wysyłaniu danych, których aplikacja się nie spodziewa i „nie potrafi" obsłużyć. Są metody przeciwdziałania tego rodzaju atakom. W tej części rozdziału skupimy się na przeciążaniu systemów, a później zajmiemy się atakami zaburzającymi polegającymi na generowaniu błędnych danych. Czasami stos protokołów w prostszych urządzeniach może nie być w stanie przetworzyć ruchu, na który nie jest przygotowany. Do generowania takich danych służy między innymi program `fragroute`.

Program `fragroute`, którego autorem jest Dug Song, został napisany wiele lat temu. Zawiera zdefiniowane reguły, według których przetwarza wszystkie pakiety wysyłane na określony adres IP. Za pomocą tego rodzaju narzędzia można dowolnie manipulować pakietami wysyłanym przez system. Jedną z głównych funkcji programu jest dzielenie wskazanych pakietów na fragmenty, które w docelowym systemie powinny być składane w całość. Jednak nie każdy system jest w stanie przetworzyć naprawdę bardzo zniekształcone fragmenty, szczególnie gdy zachodzą one na siebie. W pakiecie IP pole identyfikatora służy do wiązania ze sobą wszystkich fragmentów. Fragmenty z tym samym identyfikatorem należą do tego samego pakietu. Inne pole, tzw. przesunięcie, określa miejsce fragmentu w całym pakiecie. Gdy pierwszy fragment zawiera na przykład bajty o numerach od 0 do 1200,

przesunięcie w następnym pakiecie jest równe 1201, co oznacza jednocześnie, że jest to kolejny fragment. Takich fragmentów o mniej więcej podobnej wielkości może być wiele, a stos protokołów w systemie docelowym powinien być w stanie wszystkie je składać w całość niczym elementy układanki. Jeżeli na przykład przesunięcie w jednym fragmencie jest równe 1150, a pakiety zawierają po 1200 bajtów, oznacza to, że kolejny fragment nakłada się na poprzedni. Stos protokołów powinien wtedy odpowiednio przetworzyć takie fragmenty, tzn. nie łączyć ich ze sobą. Niekiedy takie nakładające się fragmenty mogą powodować awarię systemu, który nie jest w stanie poradzić sobie z odbieraniem sprzecznych danych. Listing 2.3 przedstawia plik konfiguracyjny umożliwiający generowanie problematycznego ruchu za pomocą programu `fragroute`.

Listing 2.3. Konfiguracja programu fragroute

```
ip_chaff dup 7
ip_frag 64 new
drop random 33
dup random 40
order random
print
```

Pierwszy wiersz oznacza, że pomiędzy pakietami będą wysyłane ich duplikaty. Cyfra 7 jest wartością pola TTL (ang. *Time to Live*, czas życia) i oznacza siedem przeskoków. Z tego względu pakiety mogą być tracone podczas transmisji. Drugi wiersz powoduje dzielenie pakietów na fragmenty o wielkości 64 bajtów. Słowo `new` oznacza, że fragmenty będą nakładać się na siebie i częściej będą zawierały nowe dane, a rzadziej stare. Liczba 33 oznacza, że 33% pakietów będzie traconych, a liczba 40, że 40% pakietów będą stanowiły losowo wybrane duplikaty. Ponadto program `fragroute` będzie generował pakiety w dowolnej kolejności, co oznacza, że nie będą tworzyły one poprawnych sekwencji w chwili dotarcia do miejsca przeznaczenia. Ostatni wiersz powoduje, że program będzie wyświetlał szczegółowe informacje o tym, co się dzieje z pakietami. Program uruchamia się poleceniem `fragroute -f frag.rules 192.168.5.40`. w tym przypadku `frag.rules` jest nazwą pliku z regułami, a `192.168.5.40` jest adresem urządzenia, do którego będą wysyłane wymieszane pakiety. Powyższe argumenty możesz zmienić odpowiednio do konfiguracji swojego środowiska.

Użycie narzędzia takiego jak `fragroute` z określonym zestawem reguł nie powoduje, że urządzenie docelowe wykona jakąś pożyteczną operację. Nie to jest jednak ważne. Rzecz w tym, aby dowiedzieć się, czy urządzenie właściwie poradzi sobie z odbieranymi pakietami. Czy na przykład zwyczajnie je odrzuci? Czy system operacyjny będzie pracował poprawnie? To są ważne informacje. Zwykłe psucie niczego nie da. Musisz umieć udokumentować działanie systemu i wskazać, co trzeba w nim zmienić. Dokładne dokumentowanie pracy jest bardzo ważne, abyś mógł pomyślnie wykonać testy, uzyskał następne zamówienie lub działał dalej.

Uwaga etyczna

Systemy, nad którymi pracujesz, powinny być Twoją własnością lub musisz mieć pozwolenie na ich testowanie. Dotyczy to w szczególności sytuacji, w których możesz je uszkodzić lub spowodować ich awarię. Wszystko, o czym tu piszę, może do tego doprowadzić. Testowanie systemów w sytuacji, kiedy nie jesteś ich właścicielem lub nie masz pozwolenia na ich testowanie, jest co najmniej nieetyczne, a czasami nielegalne. Testy, nawet te najprostsze, zawsze mogą spowodować awarię. Musisz dostać na nie pozwolenie na piśmie, zawsze!

Po przygotowaniu pliku konfiguracyjnego możesz uruchomić program `fragroute` na komputerze, który ma generować ruch. Jeżeli oferuje on funkcję kierowania ruchem, możesz wysyłać pakiety do różnych sieci. Dzięki temu testy będziesz mógł wykonać za pomocą jednego systemu. Listing 2.4 przedstawia polecenie, którego użyłem do przetestowania obsługi fragmentacji w lokalnej sieci, oraz wyniki, jakie uzyskałem. Testy polegały po prostu na wysyłaniu zapytań ping do systemu docelowego. Równie łatwo można przeprowadzić testy polegające na wysyłaniu zapytań WWW.

Listing 2.4. Program fragroute użyty w plikiem reguł

```
root@kali:~# fragroute -f frag.rules 192.168.86.1
fragroute: ip_chaff -> ip_frag -> drop -> dup -> order -> print
192.168.86.227 > 192.168.86.1: icmp: type 8 code 0
192.168.86.227 > 192.168.86.1: icmp: type 77 code 74
192.168.86.227 > 192.168.86.1: icmp: type 8 code 0
192.168.86.227 > 192.168.86.1: icmp: type 90 code 83
192.168.86.227 > 192.168.86.1: icmp: type 8 code 0
192.168.86.227 > 192.168.86.1: icmp: type 90 code 83
192.168.86.227 > 192.168.86.1: icmp: type 102 code 77
192.168.86.227 > 192.168.86.1: icmp: type 102 code 77
192.168.86.227 > 192.168.86.1: icmp: type 8 code 0
Floating point exception
```

Ciekawym wynikiem testu jest błąd operacji zmiennoprzecinkowych (`Floating point exception`). Wystąpił on w programie `fragroute` podczas modyfikowania ruchu. W tym konkretnym przypadku wydaje się, że jest to błąd w programie. Niestety, po pojawieniu się tego błędu została przerwana komunikacja sieciowa i nie mogłem już ze swojego systemu Kali wysyłać pakietów. Cały ruch był generowany przez program `fragroute`. Gdy uległ on awarii, test został przerwany. W efekcie system operacyjny próbował wysyłać pakiety do nieistniejącego celu. Jest to jeszcze inny przykład wyniku testu.

Każda awaria będąca skutkiem wykonania testu oznacza problem z dostępnością danych. Gdy system ulegnie awarii, nikt nie będzie w stanie do niego się dostać. Jeżeli usługa ulegnie awarii, przestanie być dostępna dla użytkowników. Wykonywane w ten sposób testy są atakami DoS. Dlatego ważne jest zachowanie ostrożności ze względów etycznych, o których wspomniałem wcześniej, jak również dlatego, że istnieje realne prawdopodobieństwo, że system zostanie uszkodzony i przestaną być świadczone usługi dla klientów. Więcej na ten temat będzie za chwilę.

W prosty sposób testy obciążeniowe można wykonać za pomocą narzędzia `hping3`. Za pomocą tego wspaniałego programu można generować pakiety wprost z wiersza poleceń. Należy w tym celu określić wartości poszczególnych pól pakietu, a program utworzy je zgodnie z wymaganiami. Nie oznacza to jednak, że zawsze trzeba określać wszystkie pola. Można zdefiniować tylko te niezbędne, a program wypełni pozostałe pola nagłówka IP i nagłówka warstwy transportowej typowymi wartościami. Program `hping3` jest w stanie generować pakiety, nie czekając na odpowiedź, ani nawet nie wprowadzając okresów oczekiwania. Generuje największy możliwy ruch i wysyła pakiety z maksymalną szybkością. Listing 2.5 przedstawia wynik użycia tego narzędzia.

Listing 2.5. Generowanie ruchu za pomocą programu hping3

```
root@rosebud:~# hping3 --flood -S -p 80 192.168.86.1
HPING 192.168.86.1 (eth0 192.168.86.1): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
```

```
--- 192.168.86.1 hping statistic ---
75425 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Gdy wykonywałem powyższy test, byłem zdalnie połączony z systemem Kali. Zaraz po rozpoczęciu testu usiłowałem go przerwać, ponieważ uzyskałem wynik, który był mi potrzebny. Jednak system kontynuował wysyłanie pakietów z maksymalną szybkością i dostawał odpowiedzi. Z tego powodu nie mogłem wysłać do niego kombinacji *Ctrl+C* i przerwać działania programu `hping3`, który kontynuował wysyłanie mnóstwa pakietów przez sieć (na szczęście test wykonywałem w swojej lokalnej sieci, a nie w systemie należącym do kogoś innego). System operacyjny i sieć były bardzo obciążone, więc przez dłuższy czas nie uzyskiwałem żadnej reakcji. W powyższym przykładzie użyłem programu `hping3` do wysyłania komunikatów SYN (synchronizacyjnych) na port numer 80. Był to tzw. potop komunikatów SYN. Nie tylko testowałem odporność stosu protokołów (systemu operacyjnego) na zalew pakietów, lecz także zdolność sprzętu i systemu do reagowania na taki ruch. Testowałem w ten sposób również warstwę transportową.

System operacyjny powinien rezerwować niewielką ilość pamięci dla połączeń TCP (ang. *Transport Control Protocol*, protokół sterowania transmisją). Dawniej liczba slotów dla komunikatów inicjujących tzw. **połączenia półotwarte** nie była duża. Zakładano, że systemy chcące rozpocząć komunikację będą zachowywać się poprawnie i nawiązywać połączenia wykorzystywane później przez aplikacje. Po wyczerpaniu puli slotów system nie mógł nawiązywać połączeń, nawet inicjowanych przez uprawnionych użytkowników. Obecnie większość systemów znacznie lepiej radzi sobie z potopem pakietów SYN. Nawiązują po prostu półotwarte połączenia i przerywają je, stosując różne techniki, na przykład skracając okres, przez który takie połączenie może pozostać w tym stanie.

W powyższym teście były wysyłane komunikaty SYN (argument `-S`) na port numer 80 (`-p 80`). W procesie tzw. potrójnego uścisku dłoni powinny być odbierane komunikaty SYN/ACK. Nie został określony protokół, ponieważ wyznaczał go komunikat SYN. Tego rodzaju komunikaty są stosowane jedynie w protokole TCP. Ponadto program `hping3` został uruchomiony w trybie potopu pakietów (argument `--flood`). Ten sam efekt można osiągnąć, używając argumentów określających czas przeplotu (okres oczekiwania przed wysłaniem następnego komunikatu). Użyty tutaj sposób jest łatwiejszy do zapamiętania i bardziej czytelny.

Program `hping` miał kilka wersji, o czym świadczy cyfra 3 w nazwie. Jest ogólnie dostępny w wielu dystrybucjach systemu Linux. W niektórych systemach wywołuje się go, wpisując po prostu `hping`, a w innych trzeba dodać numer wersji, na przykład `hping2` lub `hping3`.

Testowanie niższych warstw stosu protokołów za pomocą takich narzędzi jak `hping3` może być przyczyną problemów, szczególnie w przypadku słabszych urządzeń. System Kali zawiera kilka innych narzędzi umożliwiających testowanie usług działających w wyższych warstwach stosu. Co przychodzi Ci na myśl w pierwszej chwili, gdy słyszysz słowo „internet”? Spotify? Facebook? Twitter? Instagram? Wszystkie te serwisy wykorzystują protokół HTTP, a użytkownik komunikuje się z odpowiednim serwerem WWW. Nic więc dziwnego, że teraz zajmiemy się testowaniem tego rodzaju serwerów. Jest to zupełni inny temat niż testowanie aplikacji działających na serwerze WWW, czym zajmiemy się znacznie później. Na razie będziemy sprawdzać, czy serwery WWW działają.

System Kali jest wyposażony w narzędzia do testowania również innych protokołów, między innymi SIP (ang. *Session Initiation Protocol*, protokół inicjujący sesję) i RTP (ang. *Real-time Transport Protocol*, protokół transmisji w czasie rzeczywistym), wykorzystywanych w usłudze VoIP (ang. *Voice over IP*, transmisja głosu za pomocą protokołu IP). Urządzenia i serwery komunikujące się za pomocą protokołu SIP wykorzystują polecenia podobne do stosowanych w protokole HTTP. Gdy urządzenie chce zainicjować połączenie, wysyła zapytanie `INVITE` (zaproś). Aby zapytanie dotarło do odbiorcy, musi przejść przez kilka serwerów lub urządzeń sieciowych. Ponieważ w wielu firmach usługa VoIP jest bardzo ważna, krytyczne znaczenie ma sprawdzenie, czy wszystkie urządzenia są w stanie obsłużyć dużą liczbę zapytań.

Protokół SIP może korzystać z protokołu transportowego TCP lub UDP (ang. *User Datagram Protocol*, protokół datagramów użytkownika), jednak w starszych wersjach preferowany jest ten drugi protokół. Dlatego niektóre narzędzia, szczególnie te starsze, opierają swoje działanie na protokole UDP. Nowocześniejsze obsługują protokół TCP, jak również TLS (ang. *Transport Layer Security*, protokół zabezpieczający warstwę transportową) uniemożliwiający podsłuchiwanie danych. Pamiętaj, że protokół SIP opiera się na protokole HTTP, co oznacza, że wszystkie nagłówki i inne informacje przesyła w postaci zwykłego tekstu. Różni się tym od binarnego protokołu H.323, również wykorzystywanego w usłudze VoIP, w którym przesyłanych danych nie da się odczytać bez użycia narzędzia dekodującego. Narzędzie `inviteflood` wykorzystuje protokół UDP i nie można go przełączyć w tryb protokołu TCP. Jest to jednak swego rodzaju zaleta: potop pakietów można wtedy wywołać szybciej, ponieważ nie trzeba czekać na nawiązanie połączenia. Listing 2.6 przedstawia efekt użycia narzędzia `inviteflood`.

Listing 2.6. Potop zapytań INVITE w protokole SIP

```
root@kali:~# inviteflood eth0 kilroy dummy.com 192.168.86.238 150000
inviteflood - Version 2.0
              June 09, 2006
source IPv4 addr:port   = 192.168.86.35:9
dest   IPv4 addr:port   = 192.168.86.238:5060
targeted UA             = kilroy@dummy.com
Flooding destination with 150000 packets
sent: 150000
```

Na podstawie informacji pojawiających się w terminalu można stwierdzić, co tu się dzieje. W pierwszym wierszu wskazywany jest interfejs, którego narzędzie `inviteflood` będzie używać do wysyłania pakietów. Po nazwie interfejsu znajduje się nazwa drugiego urządzenia. Ponieważ protokół SIP jest wykorzystywany w usłudze VoIP, nazwą tą może być numer telefonu. W powyższym przykładzie docelowym urządzeniem jest serwer SIP o zadanej nazwie. Po niej, w zależności od konfiguracji serwera, umieszczana jest nazwa domeny lub adres IP, jeżeli domena nie jest znana. W drugim przypadku adres jest podawany dwukrotnie, ponieważ kolejnym argumentem jest adres urządzenia docelowego. Na końcu wiersza znajduje się liczba zapytań do wysłania. W tym przykładzie wysłanie 150 000 zapytań zajęło kilka sekund, co oznacza, że serwer był w stanie obsłużyć dużą ich liczbę.

Zanim przejdziemy do innych tematów, zajmijmy się przez chwilę protokołem IPv6. Nie jest on wprawdzie powszechnie stosowany w internecie, na przykład nie można go użyć do otwarcia strony Google, ale jeszcze nadejdzie jego czas. Wspomniałem tu o Google, ponieważ jest to serwis, którego adres IPv6 jest udostępniany przez serwery DNS (ang. *Domain Name System*, system nazw domen).

Ponadto protokół ten jest dzisiaj stosowany do przesyłania danych w wewnętrznych sieciach niektórych instytucji. Choć ma już ponad 20 lat, jego wdrożeniu nie towarzyszyły takie problemy jak w przypadku protokołu IPv4 — kilka dziesięcioleci zajęło usuwanie najbardziej rażących błędów wykrytych w różnych jego implementacjach. Wszystko wskazuje na to, że pomimo wysiłku, jaki producenci systemów operacyjnych Microsoft lub Linux włożyli w ich rozwijanie i testowanie, wciąż trzeba wykonywać praktyczne, wszechstronne testy różnego rodzaju urządzeń.

System Kali zawiera nieźle wyposażone zestawy narzędzi do testowania protokołu IPv6. Są to dwa osobne zestawy, ponieważ protokół ten różni się od poprzednika nie tylko sposobem adresowania urządzeń. Pełna implementacja protokołu obejmuje adresację, konfigurację urządzeń, zabezpieczenia, transmisję rozgłoszeniową (ang. *multicast*), przesyłanie dużych datagramów, przetwarzanie ścieżek itp. Ponieważ są to funkcjonalności z różnych obszarów, do ich testowania potrzebne są osobne programy.

Protokół IPv6 funkcjonuje w sieciach lokalnych inaczej niż IPv4. Do wykrywania urządzeń nie jest wykorzystywany protokół ARP (ang. *Address Resolution Protocol*, protokół wykrywania adresów), tylko nowy, wzbogacony protokół ICMPv6 (ang. *Internet Control Message Protocol*, internetowy protokół przesyłania komunikatów sterujących). Wykorzystywany jest również protokół NDP (ang. *Neighbor Discovery Protocol*, protokół wykrywania sąsiadów) ułatwiający dołączanie systemów do lokalnej sieci na podstawie pozyskiwanych z niej informacji. Protokół ICMPv6 został wzbogacony o komunikaty *Router Solicitation* (zapytanie o router), *Router Advertisement* (zgłoszenie routera), *Neighbor Solicitation* (zapytanie o sąsiada) i *Neighbor Advertisement* (zgłoszenie sąsiada). Dzięki tym komunikatom urządzenia odnajdują się w sieci i uzyskują o niej niezbędne informacje, takie jak adresy bramy domyślnej i serwera DNS.

Za pomocą dostępnych narzędzi można testować niektóre funkcjonalności protokołu i sprawdzać, jak system będzie działał pod obciążeniem. Można również manipulować komunikatami tak, aby powodować niewłaściwe działanie systemu. Narzędzia `na6`, `ns6`, `ra6` i `rs6` umożliwiają wysyłanie dowolnie skonstruowanych opisanych wyżej komunikatów protokołu ICMPv6. Większość systemów dostarcza poprawnych informacji o sieci na podstawie swojej konfiguracji i posiadanych danych, natomiast wymienione narzędzia pozwalają wysyłać błędne komunikaty i sprawdzać, jak systemy na nie reagują. W zestawie narzędzi znajduje się również program `tcp6` umożliwiający wysyłanie dowolnych pakietów TCP i przeprowadzanie ataków wykorzystujących ten protokół.

Niezależnie od tego, jakich narzędzi używa się do wykonywania testów obciążeniowych, ważne jest, by jak najdokładniej notować wszystko, co się dzieje w sieci, aby wiedzieć, co było przyczyną ewentualnej awarii. Ważne jest również monitorowanie systemów i tworzenie dzienników.

Narzędzia do ataków DoS

Atak DoS nie jest tym samym co test obciążeniowy. Nie tylko cele obu operacji są różne, lecz także wykorzystywane w nich narzędzia. Testy obciążeniowe wykonuje się najczęściej za pomocą narzędzi programistycznych, które dostarczają informacji o wydajności systemów. Testy służą sprawdzaniu funkcjonowania programów lub systemów obciążonych dużą ilością komunikatów lub uszkodzonymi komunikatami. Tu pojawia się subtelna granica. Testy obciążeniowe mogą powodować awarie aplikacji lub systemu operacyjnego. Wtedy mamy do czynienia z atakami DoS. Ponadto podczas testów może

chwilowo i gwałtownie wzrastać obciążenie procesora i wykorzystanie pamięci. To również jest wartościowa informacja, ponieważ oznacza, że w programie są błędy, które należy poprawić. W tej części rozdziału przyjrzymy się programom specjalnie przygotowanych do wywoływania awarii usług.

Atak Slowloris

Tak jak potop komunikatów SYN może wyczerpać sloty półotwartych połączeń, tak odpowiednio przeprowadzony atak może unieruchomić serwer WWW. Aplikacje nie mają do dyspozycji nieograniczonych zasobów. Serwery mogą nawiązywać określoną liczbę połączeń, zależną od budowy aplikacji. Dlatego niektóre serwery są mniej podatne na ataki, a inne bardziej. Należy zaznaczyć, że prostsze urządzenia często mają ograniczone zasoby, tj. wielkość pamięci i moc procesora. Przypomnij sobie jakieś urządzenie, którym można zarządzać przez przeglądarkę, na przykład router wi-fi, modem, drukarka. Każde z nich jest serwerem WWW, dzięki czemu łatwiej jest nim zarządzać. Nie jest to jednak ich główne przeznaczenie. Dane urządzenie musi przede wszystkim funkcjonować jako punkt dostępu wi-fi, modem lub drukarka. Dostępne zasoby urządzenia są przeznaczone przede wszystkim na te cele.

Opisane urządzenia mogą być obiektami testów, ponieważ nie są przystosowane do nawiązywania dużej ilości połączeń. Oznacza to, że atak taki jak Slowloris może spowodować odcięcie urządzenia od sieci, przez co żaden użytkownik nie będzie mógł się z nim połączyć. Atak Slowloris polega na otwieraniu bardzo wielu połączeń z serwerem WWW. Różni się od zalewania serwera komunikatami SYN tym, że może być przeprowadzany powoli. Nie polega na wywoływaniu potopu komunikatów. Zamiast tego otwierane jest połączenie i przez długi czas wysyłane są niewielkie ilości danych. Serwer podtrzymuje połączenia dopóty, dopóki narzędzie wysyła do niego zapytania.

Slowloris nie jest jednak jedynym rodzajem ataków na serwery WWW. W ostatnich latach zostało wykrytych kilka błędów w oprogramowaniu serwerów. Innego typu atakiem jest Apache Killer, polegający na wysyłaniu nakładających się serii bajtów. Serwer, usiłując złożyć serie w całość, wyczerpuje całą dostępną pamięć. Ten mankament został wykryty w oprogramowaniu Apache w wersjach 1.x i 2.x.

W systemie Kali dostępny jest program `slowhttptest` umożliwiający przeprowadzanie czterech rodzajów ataków wykorzystujących protokół HTTP. Pierwszy z nich to opisany wyżej Slowloris, polegający na powolnym wysyłaniu nagłówków. Drugi atak to R-U-Dead-Yet, w którym powoli wysyłane są treści komunikatów. Można również przeprowadzać atak Apache Killer wymuszający przetwarzanie wadliwych komunikatów przez serwer. Te trzy ataki są przeciwieństwem opisanych wcześniej ataków zalewowych, ponieważ polegają na blokowaniu usługi poprzez wysyłanie niewielkich ilości danych. Listing 2.7 przedstawia przebieg ataku Slowloris na serwer Apache uruchomiony w lokalnym systemie Kali. Komunikaty nie były wysyłane poza system. Jak widać, po 26 sekundach atak się zakończył, ponieważ serwer nie mógł już nawiązywać następnych połączeń. Oczywiście był to bardzo prosty serwer ze skonfigurowanymi zaledwie kilkoma wątkami. Aplikacja uruchomiona na wielu serwerach przystosowanych do dużego obciążenia wytrzymałaby atak znacznie dłużej.

Listing 2.7. Wynik użycia programu slowhttptest

```
Wed Dec  5 15:03:49 2018:
        slowhttptest version 1.6
 - https://code.google.com/p/slowhttptest/ -
test type:                        SLOW HEADERS
number of connections:            50
```

```
URL:                              http://localhost/
verb:                             GET
Content-Length header value:      4096
follow up data max size:          68
interval between follow up data:  10 seconds
connections per seconds:          50
probe connection timeout:         5 seconds
test duration:                    240 seconds
using proxy:                      no proxy

Wed Dec  5 15:03:49 2018:
slow HTTP test status on 25th second:

initializing:        0
pending:             0
connected:           30
error:               0
closed:              20
service available:   YES
Wed Dec  5 15:03:50 2018:
Test ended on 26th second
Exit status: No open connections left
```

Będący celem tego ataku serwer Apache wykorzystywał do przetwarzania zapytań zaledwie kilka procesów pochodnych i wątków. Ich liczba została określona w konfiguracji serwera. Domyślnie są uruchamiane dwa procesy, maksymalnie 64 wątki i 150 procesów roboczych obsługujących zapytania, a proces pochodny może uruchamiać najwyżej 25 wątków. W chwili wyczerpania przez program `slowhttptest` puli dostępnych połączeń w systemie działały 54 procesy serwera, tj. jeden główny i 53 pochodne. Aby nawiązać połączenia niezbędne do obsługi zapytań, serwer uruchomił wiele procesów pochodnych i kilka wątków w każdym z nich. Dlatego pojawiło się tak wiele procesów. Do wykonania opisanego testu została wykorzystana najnowsza wersja serwera Apache, z czego płynie następujący wniosek: pomimo upływu lat tego rodzaju ataki wciąż mogą być skuteczne. Oczywiście, jak wspomniałem wcześniej, wynik ataku zależy przede wszystkim od architektury testowanego systemu.

Testy obciążeniowe SSL

Opisany tu atak nie polega na zajęciu dostępnego pasma transmisyjnego, tylko na przeciążeniu procesora szyfrowaniem danych. Od dłuższego czasu strony serwisów zakupowych wykorzystują protokoły SSL (ang. *Secure Sockets Layer*, warstwa bezpiecznych gniazd) i TLS (ang. *Transport Layer Security*, bezpieczeństwo warstwy transportowej) do szyfrowania komunikacji z klientami i zapewnienia jej poufności. Dzisiaj z protokołów SSL i TLS korzysta wiele różnych serwerów. Jeżeli otworzysz stronę wyszukiwarki Google, zauważysz, że jest ona domyślnie szyfrowana, podobnie jak strony dużych firm, na przykład Microsoftu czy Apple. Przy próbie użycia niezaszyfrowanego adresu URL (ang. *Uniform Resource Locator*, ujednolicony format adresowania zasobów), tj. po wpisaniu prefiksu *http://* zamiast *https://*, następuje automatyczne przełączenie na komunikację szyfrowaną.

Rzecz jednak w tym, że protokoły SSL/TLS wymagają dużych mocy obliczeniowych. Nowoczesne procesory bez problemu radzą sobie z normalnym obciążeniem wywoływanym szyfrowaniem danych, tym bardziej że dzisiejsze algorytmy szyfrowania są zoptymalizowane pod kątem możliwości procesorów. Jednak protokoły SSL/TLS w znacznym stopniu obciążają każdy serwer. Przede wszystkim komunikaty wysyłane przez serwery są duże, co oznacza, że do ich szyfrowania potrzebna jest

większa moc obliczeniowa niż w przypadku małych komunikatów wysyłanych przez klientów. Ponadto system klienta wysyła tylko kilka komunikatów co jakiś czas, a serwer musi wysyłać ich bardzo dużo do wielu klientów, jednocześnie wykorzystując liczne połączenia. Obciążenie jest powodowane głównie generowaniem kluczy niezbędnych do szyfrowania sesji.

Narzędzia dostępne w systemie Kali umożliwiają testowanie starszego typu usług i serwerów wykorzystujących do szyfrowania danych protokół SSL. Protokół ten został zastąpiony nowocześniejszymi technikami i dzisiaj jest rzadko używany, choć wciąż jest stosowany w wielu usługach. Dlatego ważne jest ich testowanie. Ostatni opisywany w tym podrozdziale program służy do przeprowadzania ataków DoS na serwery używające powyższego protokołu. Program `thc-ssl-dos` wykorzystuje to, że szyfrowanie danych wymaga użycia dużych mocy obliczeniowych, szczególnie po stronie serwera.

Listing 2.8 przedstawia wynik użycia programu `thc-ssl-dos` do ataku na serwer korzystający z protokołu SSL. Błędy w tym protokole są jednak znane od tak dawna, że implementujące go biblioteki często są z systemów usuwane. Choć została użyta starsza wersja serwera, widać wyraźnie, że nie było już możliwe nawiązanie pełnego połączenia SSL. Niemniej jednak gdybyś znalazł kiedyś serwer ze skonfigurowanym protokołem SSL będziesz mógł go przetestować pod kątem podatności na ataki DoS.

Listing 2.8. Atak DoS przeprowadzony za pomocą programu thc-ssl-DoS

```
root@kali:~# thc-ssl-dos -l 100 192.168.86.239  443 --accept

      _____________ ___  _________
     \__    ___/   |   \ \_   ___ \
       |    | /    ~    \/    \  \/
       |    | \    Y    /\     \____
       |____|  \___|_  /  \_____  /
                    \/         \/
            http://www.thc.org

          Twitter @hackerschoice

Greetingz: the french underground

Waiting for script kiddies to piss off.....................
The force is with those who read the source...
Handshakes 0 [0.00 h/s], 1 Conn, 0 Err
SSL: error:140770FC:SSL routines:SSL23_GET_SERVER_HELLO:unknown protocol
#0: This does not look like SSL!
```

Niepowodzenie powyższego testu jest oznaką jednego z wyzwań towarzyszących testowaniu bezpieczeństwa systemów: znalezienie słabego punktu może być trudne. Dlatego obecnie w atakach stosuje się techniki inżynierii społecznej, polegające na wykorzystywaniu ludzkich słabości, łatwowierności i nieświadomych reakcji. Często znajdowanie technicznych wad systemów jest trudniejsze od manipulowania ludźmi. Nie oznacza to jednak, że ataki techniczne nie są przeprowadzane, tym bardziej że co jakiś czas są wykrywane i upubliczniane słabe punkty systemów. Informacje na ten temat można znaleźć na stronach Bugtraq (*http://seclists.org/bugtraq*) oraz Common Vulnerabilities and Exposures (*http://cve.mitre.org*).

Ataki DHCP

Do testowania protokołu DHCP (ang. *Dynamic Host Configuration Protocol*, protokół dynamicznego konfigurowania hostów) służy program **DHCPig**. Przeprowadza on atak polegający na zajmowaniu wszystkich zasobów serwera DHCP. Serwer ten przydziela adresy IP i udostępnia informacje konfiguracyjne. Jego awaria w firmie stanowi poważny problem, ponieważ komputery pracowników nie mogą wtedy uzyskiwać adresów IP. Nierzadko serwer DHCP przydziela adres z długim czasem dzierżawy (czyli okresem, przez który komputer użytkownika może używać adresu bez konieczności jego odnawiania), jednak zazwyczaj czas ten jest dość krótki. Jest to ważne w przypadku, gdy użytkownicy są mobilni i podłączają się do sieci często i na krótki czas. W takiej sytuacji długie czasy dzierżawy adresów powodowałyby szybkie wyczerpanie ich puli. Jeżeli jednak czasy dzierżawy są krótkie, program taki jak DHCPig może przejąć wygasający adres IP, zanim klient zdąży go odnowić. Klient wtedy zostanie bez adresu i nie będzie mógł korzystać z sieci. Korzystanie z programu DHCPig jest bardzo proste. Polega na uruchomieniu skryptu *pig.py* napisanego w języku Python i wskazaniu interfejsu, który ma zostać użyty do przeprowadzenia ataku.

Testowanie szyfrowania

Możliwość szyfrowania danych przesyłanych przez internet istnieje od ponad 20 lat. Algorytmy szyfrowania, tak jak wszystko, co dotyczy bezpieczeństwa informacji, stanowią cel ataków. Pierwsza wersja protokołu SSL, użyta w przeglądarce Netscape w 1995 r., została wycofana z użycia z powodu wykrytych w niej błędów. Druga wersja również nie przetrwała długo, ponieważ zidentyfikowano w niej następne błędy; w 1996 r. pojawiła się trzecia wersja protokołu. Ostatecznie wszystkie wersje zostały wycofane z powodu problemów z szyfrowaniem danych.

Szyfrowanie danych przesyłanych przez sieć nie polega na zwykłym pobraniu komunikatu, zaszyfrowaniu go i przesłaniu dalej. Jest to tylko część większego procesu, w którym najbardziej wrażliwymi elementami są klucze. Zaszyfrowany komunikat ma wartość tylko wtedy, gdy można go odszyfrować, a do tego celu potrzebny jest klucz. W tym miejscu właśnie zaczynają pojawiać się problemy.

Istnieją dwie główne metody szyfrowania danych. Pierwsza to **szyfrowanie asymetryczne**, w którym jeden klucz jest wykorzystywany do szyfrowania, a drugi do deszyfrowania. Na pewno spotkałeś się z określeniem **szyfrowanie za pomocą klucza publicznego**, którego idea polega na użyciu dwóch kluczy: publicznego i prywatnego. Klucz publiczny może posiadać każdy z nas; możemy mieć również dostęp do klucza innej osoby. Jednak komunikat zaszyfrowany za pomocą klucza publicznego można odszyfrować tylko za pomocą klucza prywatnego. Oba klucze są ze sobą związane matematyczną formułą wykorzystującą duże liczby. Takie podejście wydaje się racjonalne, prawda? Problem jednak polega na tym, że szyfrowanie asymetryczne jest skomplikowane obliczeniowo.

W tym momencie pojawia się **szyfrowanie symetryczne**, w którym — jak się zapewne domyślasz — wykorzystywany jest tylko jeden klucz. Przy jego użyciu zarówno szyfruje się dane, jak i je deszyfruje. Szyfrowanie symetryczne jest mniej skomplikowane obliczeniowo, ale związane są z nim dwa problemy. Pierwszy polega na tym, że im dłuższy jest klucz, tym bardziej jest podatny na ataki, dlatego że haker może zebrać dużą ilość szyfrogramów (tekstów przetworzonych za pomocą algorytmu szyfrowania) i przeanalizować je w poszukiwaniu klucza. Gdy go znajdzie, może odczytać cały ruch zaszyfrowany przy użyciu tego klucza.

Drugi, ważniejszy problem, brzmi: w jaki sposób dwie strony mogą wejść w posiadanie klucza? Szyfrowanie danych na sens tylko wtedy, gdy klucz mają obie strony. Jak można przekazać klucz drugiej stronie, gdy nie ma z nią bezpośredniego kontaktu? A jeżeli obie strony są blisko siebie, to czy muszą szyfrować przesyłane między sobą komunikaty? Jedna strona może się spotkać z drugą w umówionym miejscu i przekazać klucz. Jeżeli jednak w przyszłości potrzebny będzie nowy klucz, to komunikacja nie będzie możliwa do momentu następnego spotkania. A im dłużej wykorzystywany jest jeden klucz, tym większe prawdopodobieństwo pojawienia się opisanego wcześniej pierwszego problemu.

Ten problem rozwiązało dwóch matematyków. Nie byli wprawdzie pierwszymi, którzy znaleźli rozwiązanie, ale za to opublikowali je jako pierwsi. Whitfield Diffie i Martin Hellman wpadli na pomysł, aby obie strony niezależnie od siebie generowały klucze na podstawie uzgodnionej liczby. Liczbę tę mogą sobie bezpiecznie przekazać w niezaszyfrowanej postaci, ponieważ ważne jest jedynie to, co dzieje się z nią później. Obie strony umieszczają liczbę w formule matematycznej wraz z inną, znaną tylko im wartością. Jak poprzednio, formuła też może być publiczna, ponieważ ważna jest tylko owa poufna wartość. Strony wymieniają wyniki obliczeń między sobą i ponownie stosują formułę. W ten sposób wykonują te same obliczenia, wychodząc z tego samego punktu i w efekcie uzyskują te same klucze.

Powyższy sposób stosuje się w praktyce dlatego, że wykorzystywane są w nim wszystkie opisane wcześniej mechanizmy. Klucz Diffiego-Hellmana jest stosowany w kryptografii do generowania symetrycznego klucza sesji. Dzięki temu obliczenia są mniej skomplikowane i serwer jest w mniejszym stopniu obciążany szyfrowaniem i deszyfrowaniem dużych ilości danych przesyłanych pomiędzy nim a klientami.

Jak wspomniałem wcześniej, protokół SSL nie jest już stosowany w kryptografii. Obecnie wykorzystuje się protokół TLS. Pojawiło się już kilka wersji tego protokołu, co świadczy o wyzwaniach towarzyszących szyfrowaniu danych. Obecnie stosowana jest wersja 1.2, a w przygotowaniu jest wersja 1.3. Kolejne wersje zawierają poprawki błędów znajdowanych podczas nieustających prób złamania tego protokołu.

Do sprawdzenia, czy serwer wykorzystuje przestarzały protokół, służy między innymi narzędzie `sslscan`. Ten program usiłuje określić algorytm szyfrowania używany przez serwer. Jest to proste zadanie, ponieważ podczas nawiązywania połączenia serwer wysyła do klienta listę szyfrów do wyboru. Zatem aby program `sslscan` uzyskał wszystkie potrzebne informacje, musi jedynie zainicjować sesję. Listing 2.9 przedstawia wynik testu algorytmu szyfrowania wykorzystywanego przez serwer Apache.

Listing 2.9. Test lokalnego serwera za pomocą programu sslscan

```
root@kali:~# sslscan 192.168.1.1
Version: 1.11.12-static
OpenSSL 1.0.2-chacha (1.0.2g-dev)

Connected to 192.168.1.1

Testing SSL server 192.168.1.1 on port 443 using SNI name 192.168.1.1

  TLS Fallback SCSV:
Server supports TLS Fallback SCSV
```

```
  TLS renegotiation:
Secure session renegotiation supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLS 1.2 not vulnerable to heartbleed
TLS 1.1 not vulnerable to heartbleed
TLS 1.0 not vulnerable to heartbleed

  Supported Server Cipher(s):
Preferred TLSv1.2  128 bits  ECDHE-RSA-AES128-GCM-SHA256   Curve P-256 DHE 256
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-GCM-SHA384   Curve P-256 DHE 256
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA          Curve P-256 DHE 256
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA          Curve P-256 DHE 256
Accepted  TLSv1.2  128 bits  AES128-GCM-SHA256
Accepted  TLSv1.2  256 bits  AES256-GCM-SHA384
Accepted  TLSv1.2  128 bits  AES128-SHA
Accepted  TLSv1.2  256 bits  AES256-SHA
Accepted  TLSv1.2  112 bits  DES-CBC3-SHA
Preferred TLSv1.1  128 bits  ECDHE-RSA-AES128-SHA          Curve P-256 DHE 256
Accepted  TLSv1.1  256 bits  ECDHE-RSA-AES256-SHA          Curve P-256 DHE 256

  SSL Certificate:
Signature Algorithm: sha256WithRSAEncryption
RSA Key Strength:    2048

Subject:  kali
Issuer:   kali

Not valid before: Dec  5 08:59:00 2018 GMT
Not valid after:  Dec  5 08:59:00 2019 GMT
```

Program `sslscan` sprawdza, czy w protokole wykorzystywanym przez serwer jest błąd Heartbleed. W wyniku ataku wykorzystującego tę lukę możliwe jest uzyskanie kluczy niezbędnych do szyfrowania komunikacji pomiędzy serwerem a klientem. Ponadto program wyświetla listę obsługiwanych szyfrów. Składa się ona z kilku kolumn, w większości zawierających niezbyt zrozumiałe informacje. Pierwsza kolumna jest dość czytelna: informuje, czy dany protokół i szyfr są preferowane czy tylko akceptowane przez serwer. Zauważ, że każda wersja protokołu TLS zawiera własny preferowany szyfr. W drugiej kolumnie znajdują się nazwa i wersja protokołu. Ten serwer w ogóle nie obsługuje protokołu SSL, ponieważ zostały usunięte implementujące go biblioteki. Następna kolumna zawiera informację o sile klucza.

Siły kluczy, które są stosowane w różnych algorytmach, nie można porównywać na podstawie ich długości. Algorytm RSA (Rivest-Shamir-Adleman) jest algorytmem asymetrycznym, w którym długość klucza jest wielokrotnością liczby 1024. Algorytm AES (ang. *Advanced Encryption Standard*, zaawansowany standard szyfrowania) jest algorytmem symetrycznym, w którym klucz ma długość 128 lub 256 bitów. Nie oznacza to, że RSA jest algorytmem kilka rzędów wielkości silniejszym niż AES, tylko że klucze są wykorzystywane w inny sposób. Porównywanie algorytmu symetrycznego z asymetrycznym również jest mylące, ponieważ w każdym z nich klucze są wykorzystywane zupełnie inaczej.

Następna kolumna zawiera **zestaw szyfrów**. Określenie to wzięło się stąd, że w całym procesie szyfrowania jest wykorzystywanych do różnych celów kilka algorytmów. Weźmy dla przykładu zestaw DHE-RSA-AES256-SHA256. Pierwszy człon nazwy, DHE, oznacza, że do wymiany kluczy jest wykorzystywany algorytm Diffie-Hellman Ephemeral. Druga część, RSA, jest skrótem od nazwisk autorów algorytmu, Rivest-Shamir-Adleman. Jest to algorytm asymetryczny wykorzystywany do uwierzytelniania stron. Klucze są przechowywane w certyfikatach, które również zawierają informacje identyfikacyjne serwera. Jeżeli certyfikat ma również klient, strony uwierzytelniają się wzajemnie. W przeciwnym wypadku klient uwierzytelnia serwer na podstawie uzyskanej od niego nazwy oraz nazwy zapisanej w certyfikacie. Algorytm asymetryczny jest wykorzystywany także do szyfrowania kluczy przesyłanych między klientem a serwerem.

W opisie bardzo często pojawiają się słowa *serwer* i *klient*, dlatego ważne jest, abyś wiedział, co one oznaczają. Komunikacja sieciowa zawsze odbywa się pomiędzy klientem a serwerem. Nie oznacza to jednak, że serwer jest urządzeniem znajdującym się w centrum danych, tylko że udostępnia określoną usługę. Zawsze to klient jest stroną rozpoczynającą komunikację, a serwer stroną odpowiadającą. Można łatwo sprawdzić, która strona inicjuje połączenie, a która odpowiada.

Następna kolumna to nazwa algorytmu szyfrowania symetrycznego. Można się domyśleć, że w algorytmie AES jest wykorzystywany klucz o długości 256 bitów. Warto zaznaczyć, że AES w rzeczywistości nie jest algorytmem, tylko standardem szyfrowania. Algorytmy spełniające ten standard mają własne nazwy. Przez dziesięciolecia był stosowany standard DES (ang. *Data Encryption Standard*, standard szyfrowania danych), wykorzystujący szyfr Lucifer opracowany w firmie IBM przez Horsta Feistela i jego zespół. W latach 90. ubiegłego wieku stwierdzono, że standard ten jest już przestarzały i wkrótce zostanie złamany. Rozpoczęto więc prace nad standardem AES, w którym podstawą był algorytm Rijndaela. Początkowo wykorzystywany był w nim klucz o długości 128 bitów. Klucz o długości 256 bitów jest powszechnie stosowany od dość niedawna.

AES jest standardem szyfrowania sesji. Wykorzystywany w nim 256-bitowy klucz jest generowany i przekazywany drugiej stronie na początku sesji. Jeżeli sesja trwa długo, klucz może zostać wygenerowany ponownie, aby uniknąć jego przejęcia w wyniku ataku. Jak wspomniałem wcześniej, klucz jest wykorzystywany przez obie strony zarówno do szyfrowania, jak i deszyfrowania danych.

Zapewne zauważyłeś w listingu oznaczenie algorytmu `SHA256` (ang. *Secure Hash Algorithm*, algorytm bezpiecznego kodowania). Wykorzystywany jest w nim klucz o długości 256 bitów. Algorytm ten służy do sprawdzania, czy dane nie zostały zmienione. Znasz pewnie algorytm MD5 (ang. *Message Digest*, skrót wiadomości). Oba algorytmy różnią się długością generowanych skrótów. W algorytmie MD5 skrót zawsze zajmuje 32 bajty, z których wykorzystywanych jest tylko 128 bitów (po cztery bity w każdym bajcie). Algorytm ten został zastąpiony algorytmem SHA1 i jego nowszymi wersjami. W wersji SHA1 skrót zajmuje 40 bajtów i wykorzystuje 160 bitów (jak poprzednio: tylko 4 bity w bajcie). W powyższym listingu widać algorytm SHA256 generujący skrót o długości 64 bajtów. Niezależnie od tego, ile jest przesyłanych danych, skrót zawsze ma taką samą długość. Jest on przesyłany drugiej stronie, która na tej podstawie sprawdza, czy odebrane przez nią dane nie zostały zmienione. Jeżeli w danych zostanie zmieniony choćby jeden bit, skrót uzyskany za pomocą algorytmu SHA lub MD5 będzie inny.

Wszystkie wymienione algorytmy współpracują ze sobą i tworzą protokół TLS (następcę SSL). Do skutecznego szyfrowania danych i ochrony ich przed przechwyceniem niezbędny jest komplet algorytmów. Każda strona musi być w stanie wygenerować klucz sesji. Przed wygenerowaniem klucza obie strony muszą się wzajemnie uwierzytelnić i przekazać sobie informacje o szyfrowaniu. Do szyfrowania i deszyfrowania danych przesyłanych w trakcie sesji potrzebny jest klucz i algorytm. Ponadto strony muszą sprawdzać, czy dane nie zostały zniekształcone. Jest to przykład użycia zestawu algorytmów zapewniającego silne szyfrowanie danych.

Gdyby w informacjach wyświetlonych przez program pojawił się algorytm 3DES oznaczałoby to, że serwer jest podatny na ataki mające na celu przechwycenie klucza. Wtedy dane mogłaby odszyfrować osoba nieuprawniona. Ponadto — o czym wspomniałem wcześniej — program `sslscan` sprawdza, czy wykorzystywane przez serwer protokoły są podatne na ataki przy użyciu znanych eksploitów.

W rzadkich przypadkach w miejscu skrótu AES256 może pojawić się NULL, oznaczający, że szyfrowanie nie jest stosowane. Może tak być z kilku powodów. Ochrona przesyłanych danych może nie być istotna, za to ważne jest sprawdzenie, jakie strony komunikują się ze sobą oraz czy dane nie są zmieniane podczas transmisji. Dlatego klient nie żąda szyfrowania, aby nie obciążać systemu, natomiast chce korzystać z funkcjonalności oferowanych przez inne algorytmy znajdujące się w zestawie.

Rozwój technik kryptograficznych trwa cały czas. Nawet dzisiaj prowadzone są badania mające na celu wykrycie słabych punktów w algorytmach szyfrujących i protokołach. W miarę upływu czasu, gdy będą pojawiały się nowe klucze i coraz silniejsze algorytmy, będziesz zauważał różnice pojawiające się w wynikach wyświetlanych przez opisany program.

Przechwytywanie pakietów

Podczas testowania sieci warto jest wiedzieć, jakie dane są przez nią przesyłane. Do tego celu potrzebny jest program przechwytujący pakiety, a mówiąc ściślej — ramki. Wspominam o tej różnicy dlatego, że w każdej warstwie stosu protokołów sieciowych stosowane są inne określenia porcji danych. Pamiętaj, że podczas przemieszczania się danych w dół stosu dodawane są do nich kolejne nagłówki. Ostatni nagłówek jest dodawany w warstwie drugiej, w której jednostka PDU (ang. *Protocol Data Unit*, jednostka danych protokołu) nosi nazwę *ramki*. W warstwie trzeciej jest to *pakiet*, a w czwartej *datagram* lub *segment*, w zależności od stosowanego protokołu.

Wiele lat temu przechwytywanie pakietów było kosztowną operacją, ponieważ wymagało użycia specjalnego interfejsu sieciowego, który mógł pracować w trybie nasłuchiwania (ang. *promiscuous mode*). Było to niezbędne, gdyż domyślnie interfejsy sieciowe sprawdzają adresy MAC odbieranych ramek. Interfejs „zna" własny, identyfikujący go adres MAC. Jeżeli adres odebranej ramki jest zgodny z adresem interfejsu lub jest to adres rozgłoszeniowy, ramka zostaje przekazana do systemu operacyjnego. Wszystkie inne ramki zostają odrzucone. Natomiast interfejs działający w trybie nasłuchu przekazuje do systemu operacyjnego wszystkie ramki, niezależnie od tego, czy są one dla niego przeznaczone, czy nie. Z całą pewnością jest to cenna funkcjonalność, ale o wiele cenniejsza jest możliwość zaglądania we wszystkie ramki trafiające do interfejsu.

Dzisiejsze interfejsy sieciowe umożliwiają nie tylko automatyczne negocjacje parametrów i w pełni dupleksową transmisję, lecz także mogą pracować w trybie nasłuchu. Oznacza to, że nie trzeba już

stosować analizatorów protokołów (jak jest często nazywany sprzęt z interfejsem o opisanej wyżej funkcjonalności), ponieważ każdy system może być takim analizatorem. Aby podejrzeć przesyłane dane, wystarczy jedynie wiedzieć, jak przechwytywać ramki i interpretować ich zawartość.

Program tcpdump

Niektóre systemy operacyjne mają własne programy do rejestrowania pakietów. Na przykład w systemie Solaris jest to program `snoop`. Jednak dzisiaj, szczególnie w systemach Linux, stosowany jest niemal wyłącznie program `tcpdump`, z którego korzysta się w wierszu poleceń. W dalszej części rozdziału opisany jest inny, graficzny program, ale i tak bardzo cenna jest znajomość podstawowego narzędzia. Bardzo często będziesz miał do dyspozycji jedynie terminal i połączenie SSH umożliwiające uruchamianie programów wyłącznie za pomocą wiersza poleceń. Dlatego warto jest znać program `tcpdump`. Kiedyś na przykład użyłem go do sprawdzenia, czy protokół SIP rzeczywiście korzysta z protokołu UDP, a nie TCP. Wiedza o wykonywanych przez aplikacje operacjach, które nie są widoczne wprost, jest naprawdę cenna.

Zanim zajmiemy się opcjami programu `tcpdump`, przyjrzyjmy się wyświetlanym przez niego informacjom. Nauczenie się tego, jak je interpretować, zajmuje nieco czasu. Po uruchomieniu programu bez żadnych opcji pojawia się podsumowanie wszystkich przechwyconych pakietów. Listing 2.10 przedstawia przykładowy wynik użycia programu `tcpdump`.

Listing 2.10. Wynik użycia programu tcpdump

```
root@kali:~# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode listening on eth0,
↪link-type EN10MB (Ethernet), capture size 262144 bytes
10:26:26.543550 IP binkley.lan.57137 > testwifi.here.domain: 32636+ PTR?
c.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.f.f.ip6.arpa. (90)
10:26:26.555133 IP testwifi.here.domain > binkley.lan.57137: 32636 NXDomain 0/1/0 (154)
10:26:26.557367 IP binkley.lan.57872 > testwifi.here.domain: 44057+ PTR? 201.86.168.192.
↪in-addr.arpa. (45)
10:26:26.560368 IP testwifi.here.domain > binkley.lan.57872: 44057* 1/0/0 PTR  kilroyhue.lan. (99)
10:26:26.561678 IP binkley.lan.57726 > testwifi.here.domain: 901+ PTR? 211.1.217.172.
↪in-addr.arpa. (44)
10:26:26.583550 IP testwifi.here.domain > binkley.lan.57726: 901 4/0/0 PTR
↪den16s02-in-f19.1e100.net., PTR iad23s26-in-f211.1e100.net., PTR den16s02-in-f19.1e100.net.,
↪PTR iad23s26-in-f211.1e100.net. (142)
10:26:26.585725 IP binkley.lan.64437 > testwifi.here.domain: 23125+ PTR? 0.0.0.0.in-addr.arpa. (38)
10:26:26.598434 IP testwifi.here.domain > binkley.lan.64437: 23125 NXDomain 0/1/0 (106)
10:26:26.637639 IP binkley.lan.51994 > 239.255.255.250.ssdp: UDP, length 174
```

Pierwsza kolumna w powyższym listingu zawiera znacznik czasu, czyli bieżący czas podawany z dokładnością do ułamka sekundy. Nie jest to informacja pozyskiwana bezpośrednio z pakietu, ponieważ żaden z nagłówków pakietu nie przekazuje znacznika czasu. Druga kolumna zawiera nazwę protokołu z warstwy sieciowej. Nie ma informacji o protokole z warstwy danych, ponieważ jest on zdeterminowany przez typ interfejsu sieciowego i ta informacja jest oczywista. Najczęściej jest to Ethernet.

Kolejne dane reprezentują komunikujące się strony. Są to nie tylko nazwy komputerów, lecz także numery portów. W tym przypadku nadawcą pierwszego pakietu jest komputer o nazwie `binkley.lan`, a odbiorcą `testwifi.here`. Program `tcpdump` zamienia adresy IP stron na ich nazwy. Aby wyłączyć tę funkcję, należy program uruchomić z argumentem `-n`. Pakiety można wtedy przechwytywać z więk-

szą prędkością, ponieważ system nie musi odpytywać serwera DNS o nazwy wszystkich komputerów wymieniających między sobą dane.

Zapewne zauważyłeś, że w nazwie komputera znajduje się liczba. Jest to numer portu. Przykładowo w nazwie nadawcy `binkley.lan.57137` oznacza ona port źródłowy. Nazwa odbiorcy `testwifi.here.domain` zawiera sufiks `domain` oznaczający, że użyty jest standardowy port serwera DNS. Podobnie jak to było w przypadku adresów IP i nazw komputerów, program `tcpdump` nie musi zamieniać standardowego numeru portu na jego nazwę. Jeżeli użyje się argumentu `-n`, wyświetlany będzie po prostu numer portu, w tym przypadku 53.

Listing 2.10 przedstawia głównie zapytania i odpowiedzi serwera DNS, które są efektem wyszukiwania przez program `tcpdump` nazw przypisanych adresom IP. Końcówka każdego wiersza to opis pakietu. Jeżeli pakiet zawiera komunikat protokołu TCP, to w opisie znajdują się oznaczenia flag z nagłówka oraz numer sekwencyjny.

Teraz przyjrzyjmy się wynikom, które można uzyskać, uruchamiając program `tcpdump` z argumentem `-v`. Program dostarcza informacji o różnym poziomie szczegółowości w zależności od zastosowanego argumentu. Sprawdźmy również efekt użycia argumentu `-n` wyłączającego zamianę adresów IP na nazwy. Listing 2.11 przedstawia bardziej szczegółowy wynik działania programu.

Listing 2.11. Dokładniejsze informacje dostarczane przez program tcpdump

```
root@kali:~# tcpdump -v -n
tcpdump: listening on eth0, link-type EN10MB (Ethernet),
↪capture size 262144 bytes
11:39:09.703339 STP 802.1d, Config, Flags [none], bridge-id 7b00.18:d6:c7:7d:f4:8a.8004,
↪length 35 message-age 0.75s, max-age 20.00s, hello-time 1.00s, forwarding-delay 4.00s
↪root-id 7000.2c:08:8c:1c:3b:db, root-pathcost 4
11:39:09.710628 IP (tos 0x0, ttl 233, id 12527, offset 0, flags [DF], proto TCP (6),  length 553)
   54.231.176.224.443 > 192.168.86.223.62547: Flags [P.], cksum 0x6518 (correct), seq
   ↪3199:3712, ack 1164, win 68, length 513
11:39:09.710637 IP (tos 0x0, ttl 233, id 12528, offset 0, flags [DF], proto TCP (6),  length 323)
   54.231.176.224.443 > 192.168.86.223.62547: Flags [P.],  cksum 0x7f26 (correct), seq
   ↪3712:3995, ack 1164, win 68, length 283
11:39:09.710682 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6),  length 40)
   192.168.86.223.62547 > 54.231.176.224.443: Flags [.], cksum 0x75f2 (correct),
   ↪ack 3712, win 8175, length 0
11:39:09.710703 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6),  length 40)
```

Wyświetlone informacje są bardzo podobne do uzyskanych w poprzednim przykładzie. Różnica polega na tym, że zamiast nazw komputerów i portów pojawiły się odpowiadające im liczby. Jest to efekt użycia argumentu `-n`. W listingu widoczne są adresy IP obu stron i numery portów. Dzięki argumentowi `-v` uzyskuje się dokładniejsze informacje o nagłówkach pakietów. Ponadto sprawdzana jest suma kontrolna, a wynik weryfikacji jest oznaczany słowem `correct` lub `incorrect`. Widoczne są również wartości innych pól nagłówków, między innymi TTL i identyfikatory pakietów.

Nawet jeżeli za pomocą argumentu `-vvv` włączy się tryb najwyższej szczegółowości, nie uzyska się pełnych informacji umożliwiających analizę zawartości pakietów. Przechwycone pakiety można jednak zapisać w pliku. W tym miejscu należy wspomnieć o długości fragmentu (ang. *snap*). Jest to liczba przechwytywanych bajtów w każdym pakiecie. Domyślnie są to 262144 bajty, ale tę liczbę można zmniejszyć. Należy pamiętać, że wartość 0 oznacza, że program `tcpdump` będzie rejestrował maksymalną, wymienioną wyżej liczbę bajtów. Aby zapisać pakiety, trzeba użyć argumentu `-w` i podać nazwę

pliku. Zostanie wtedy utworzony plik w formacie *.pcap*, który można zaimportować do programu odczytującego tego rodzaju pliki. Za chwilę poznasz jeden z nich.

Filtr BPF

Inną ważną funkcjonalnością programu `tcpdump`, którą za chwilę zastosujemy, jest filtr BPF (ang. *Berkeley Packet Filter*, filtr pakietów opracowany na uniwersytecie Berkeley). Składa się on z pól i parametrów definiujących pakiety, które mają być przechwytywane. Jeżeli ruch w sieci jest duży, to przechwytywanie wszystkich pakietów może doprowadzić do szybkiego zapełnienia dysku komputera. Jeśli zawczasu wiadomo, jakie pakiety są potrzebne, da się zdefiniować odpowiedni filtr. Ponadto dzięki temu można również ułatwić sobie analizę wyświetlanych informacji i zaoszczędzić mnóstwo czasu.

Najbardziej podstawowy filtr opisuje protokół sieciowy. Można na przykład rejestrować pakiety przesyłane wyłącznie za pomocą protokołu TCP, UDP, IP lub dowolnego innego. Listing 2.12 przedstawia wynik przechwycenia pakietów przesłanych tylko przy użyciu protokołu ICMP. Pamiętaj, że definicję filtru umieszcza się na końcu polecenia. Program `tcpdump` analizuje wszystkie pakiety odbierane i wysyłane przez interfejs sieciowy, a jeżeli są to żądane pakiety, wyświetla je w terminalu lub zapisuje w pliku.

Listing 2.12. Wynik użycia programu tcpdump z filtrem BPF

```
root@kali:~# tcpdump icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
12:01:14.602895 IP binkley.lan > rosebud.lan: ICMP echo request, id 8203, seq 0,  length 64
12:01:14.602952 IP rosebud.lan > binkley.lan: ICMP echo reply, id 8203, seq 0,  length 64
12:01:15.604118 IP binkley.lan > rosebud.lan: ICMP echo request, id 8203, seq 1,  length 64
12:01:15.604171 IP rosebud.lan > binkley.lan: ICMP echo reply, id 8203, seq 1,  length 64
12:01:16.604295 IP binkley.lan > rosebud.lan: ICMP echo request, id 8203, seq 2,  length 64
```

W filtrach można stosować również wyrażenia logiczne. Za pomocą operatorów można tworzyć skomplikowane filtry. Załóżmy, że interesują nas pakiety wysyłane i odbierane przez serwer WWW. Aby wyświetlić te pakiety, należy użyć na przykład filtru `tcp and port 80`, który przepuszcza wszystkie pakiety TCP zawierające port numer 80. Zauważ, że wskazany port nie został określony jako źródłowy ani docelowy, ale oczywiście można to zrobić, używając filtru `src port 80` lub `dst port 80`. Jeżeli nie określi się rodzaju portu, to przechwytywane są pakiety przesyłane w obie strony. Pakiet wysyłany do serwera WWW ma port docelowy numer 80 i pewien port źródłowy. Gdy serwer wysyła odpowiedź do klienta, zamienia porty miejscami. Źródłowy port ma wtedy numer 80. Jeżeli zostanie zastosowany filtr `src port 80`, to będą przechwytywane tylko pakiety przesyłane w jednym kierunku. Oczywiście mogą to być właśnie te pakiety, które nas interesują. Pamiętaj jednak o zasadzie określania portów w filtrze. Można również określać zakresy portów, na przykład `80-88`.

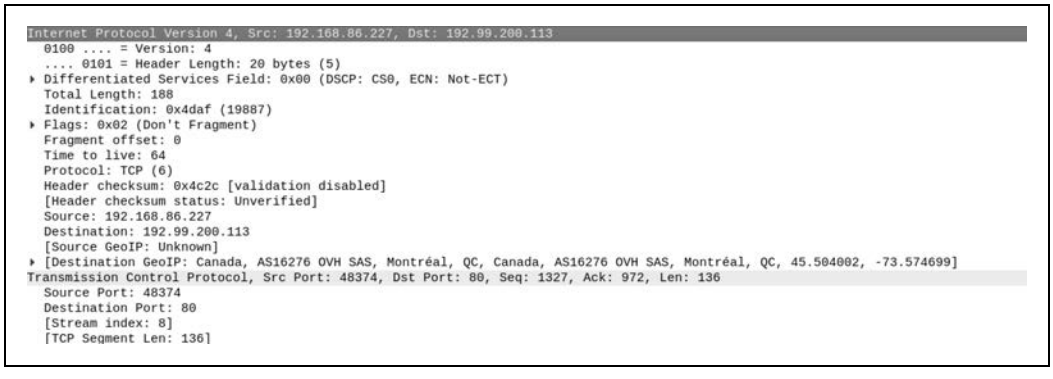
Składnia filtra BPF oferuje mnóstwo możliwości. Jeżeli potrzebny jest naprawdę zaawansowany filtr, można oczywiście zagłębić się w szczegóły składni BPF lub przeanalizować przykładowe filtry, podobne do żądanego. Z mojego doświadczenia wynika, że często pojawia się potrzeba określenia portu. Często jest również znany adres komputera wysyłającego i odbierającego pakiety. Wtedy trzeba użyć filtru na przykład `host 192.168.86.35`, który przepuszcza wyłącznie pakiety z podanym adresem IP źródłowym lub docelowym. Tak jak poprzednio w powyższym filtrze adres nie został określony

jako źródłowy ani docelowy. Można to jednak zrobić, stosując filtr `src host` lub `dst host`. Jeżeli rodzaj adresu nie zostanie określony, to przechwytywane będą pakiety przesyłane w obu kierunkach.

Nawet podstawowa znajomość składni filtra BPF pozwoli Ci wybierać żądane dane i skupiać się na ich analizie. Każdy pakiet zawiera mnóstwo szczegółowych informacji i gdy zaczniesz je analizować, przekonasz się, jak żmudne jest to zadanie.

Wireshark

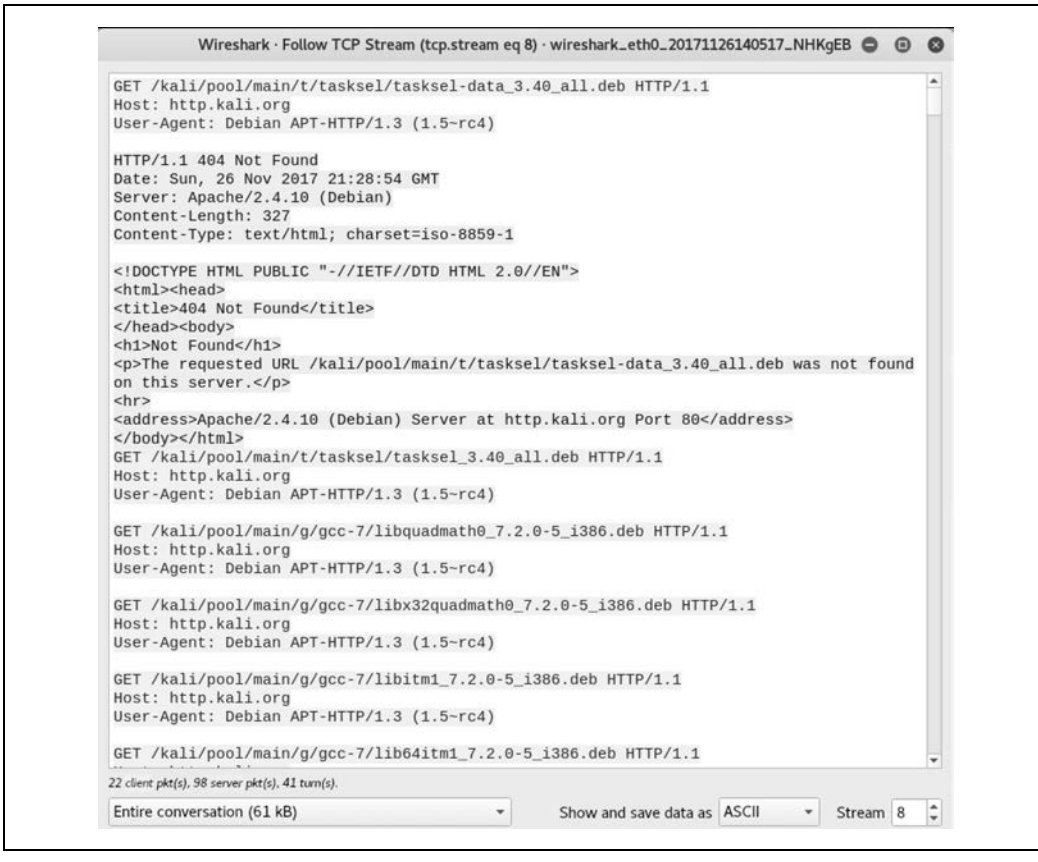
Kiedy już zapiszesz pakiety w pliku, warto byłoby, abyś je przeanalizował. Jednym z najlepszych narzędzi służących do tego celu jest Wireshark. Oczywiście program ten również pozwala przechwytywać pakiety i zapisywać je w plikach *.pcap* do późniejszej analizy. Jednak podstawową zaletą programu Wireshark jest możliwość naprawdę głębokiego wnikania w szczegóły pakietów. Zamiast trwonić czas na poznawanie wyglądu i działania programu, przejdźmy od razu do analizowania pakietów. Rysunek 2.4 przedstawia informacje o nagłówkach IP i TCP pakietu HTTP.



Rysunek 2.4. Pola nagłówków pakietu widoczne w programie Wireshark

Jak widać na powyższym rysunku, Wireshark dostarcza więcej szczegółowych informacji niż `tcpdump`. Jest to jedna z wielkich zalet interfejsu graficznego. W oknie programu jest po prostu więcej miejsca, dzięki czemu dane z nagłówków mogą być prezentowane w bardziej czytelny sposób. Każde pole nagłówka jest wyświetlane w osobnym wierszu i widać wyraźnie, jakie informacje zawiera. Co więcej, niektóre pola, na przykład z flagami, są wyświetlane w kilku wierszach i prezentowanych jest jeszcze więcej szczegółów. Wymienione pole składa się z serii bitów. Aby zobaczyć ich wartości, wystarczy kliknąć symbol trójkąta i rozwinąć dodatkowe wiersze. Ponadto informacje te są widoczne w głównym wierszu, ponieważ program Wireshark wyświetla ich skrót. W tym przypadku ustawiony jest bit DF (ang. *Don't Fragment*, nie fragmentuj).

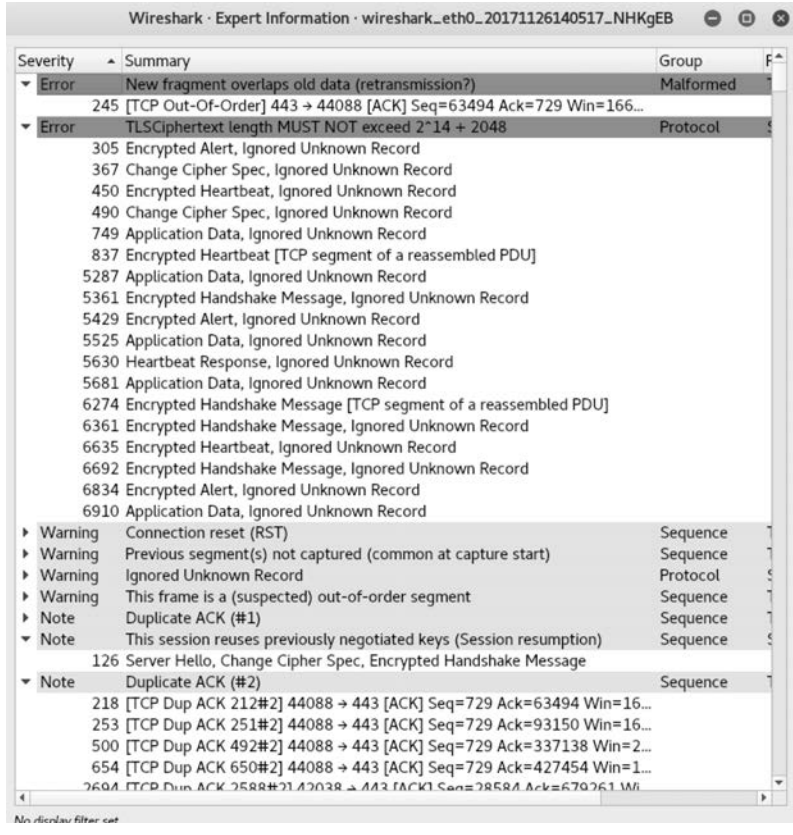
Inną zaletą narzędzia takiego jak Wireshark jest łatwiejszy wgląd w zawartość pakietu. Aby znaleźć pakiet będący częścią przesyłanych danych, które nas interesują, wystarczy wybrać polecenie *Follow/TCP Stream* (podążaj/za strumieniem TCP). Uzyska się wtedy nie tylko informacje o pakietach tworzących żądaną konwersację, lecz także wgląd do zawartości pakietów, która w oknie jest wyświetlana w formacie ASCII. Przedstawia to rysunek 2.5. Ponadto dane w programie Wireshark są wyróżniane kolorami. W tym przypadku kolor czerwony oznacza pakiety wysłane przez klienta, a niebieski przez serwer. W dolnej części okna znajduje się również krótkie podsumowanie liczby pakietów wysłanych przez obie strony.



Rysunek 2.5. Wynik użycia polecenia Follow TCP Stream

Program Wireshark oferuje te same funkcjonalności filtrowania pakietów co `tcpdump`. Filtry można stosować w trakcie lub po zakończeniu przechwytywania pakietów, gdy są już wyświetlane na ekranie. Program w dużej mierze wspomaga użytkownika w definiowaniu filtrów. Podczas wpisywania znaków w polu filtru w górnej części okna program stara się uzupełniać polecenia. Informuje również, czy filtr jest poprawny. Błąd jest sygnalizowany czerwonym kolorem, a poprawny filtr kolorem zielonym. Można stosować wszystkie pola i cechy pakietu, które Wireshark zna, i na przykład filtrować pakiety zawierające określony kod odpowiedzi HTTP. Jest to cenna funkcjonalność, przydatna podczas diagnozowania przyczyn problemów z komunikacją sieciową.

Ponadto Wireshark przeprowadza mnóstwo różnego rodzaju analiz. Przykładowo: pakiety podzielone na fragmenty za pomocą opisanego wcześniej programu `fragroute` oznacza kolorem sygnalizującym, że coś jest z nimi nie w porządku. Jeżeli nie zgadza się suma kontrolna, pakiet jest oznaczany kolorem czarnym. Wszelkie błędy protokołu powodujące zniekształcenia pakietów, jak również zerwane połączenia TCP, są sygnalizowane kolorem czerwonym. Kolor żółty oznacza ostrzeżenie, na przykład o nietypowym kodzie błędu zgłaszanym przez aplikację lub o problemie z połączeniem. Nieco czasu można zaoszczędzić, korzystając z menu *Analyze* (analiza) i opcji *Expert Info* (informacje specjalistyczne), wyświetlającej między innymi pełną listę pakietów podzielonych na fragmenty. Rysunek 2.6 przedstawia przykładowy widok.



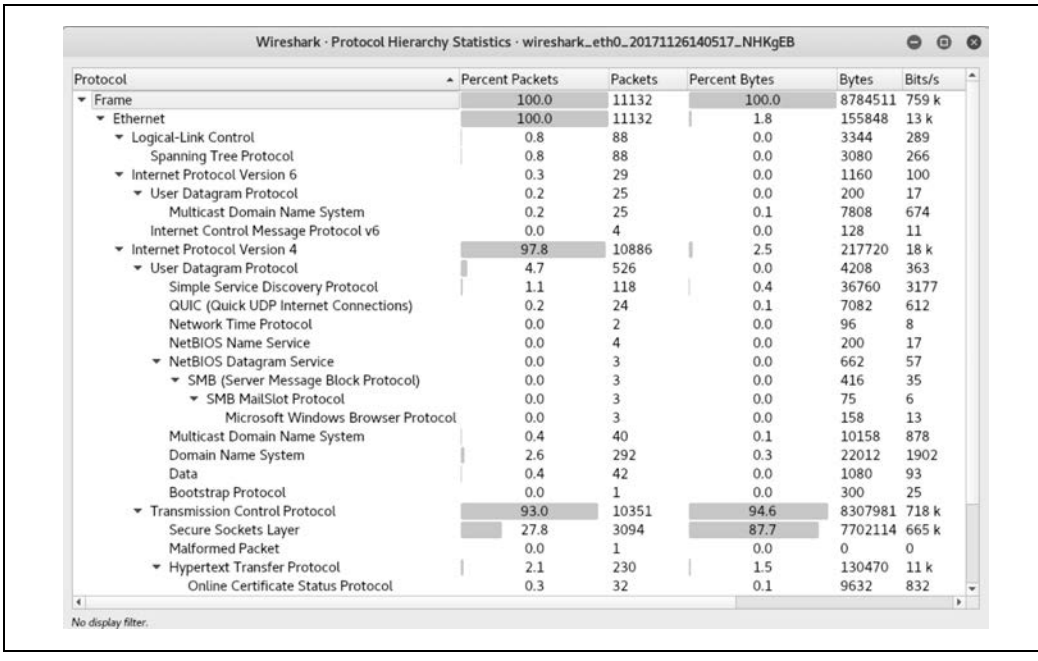
Rysunek 2.6. Wynik użycia opcji Expert Info

Opisane wyżej funkcjonalności programu Wireshark stanowią zaledwie drobny fragment jego możliwości. Mnóstwo przydatnych informacji zawartych w nagłówkach tworzonych przez poszczególne protokoły jest w czytelny sposób podzielonych na osobne wiersze. Dzięki temu można łatwo sprawdzać na przykład przyczyny błędów pojawiających się podczas testów.

Inną przydatną funkcjonalnością programu Wireshark jest menu *Statistics* (statystyki). Służy ono między innymi do prezentowania danych o przechwyconych projektach w formie graficznej, chociażby w postaci wykresów. Rysunek 2.7 przedstawia przykładowy widok hierarchii protokołów.

Widok ten umożliwia między innymi szybką identyfikację nieznanych lub najczęściej stosowanych protokołów. Jeżeli podejrzewasz, że miał miejsce atak oparty na protokole UDP, ale udział pakietów przesłanych za pomocą tego protokołu był niewielki, musisz dokonać głębszej analizy.

Program Wireshark został zainstalowany domyślnie w systemie Kali, ale jest dostępny również dla innych systemów, na przykład Windows, macOS i różnych dystrybucji Linuxa. Nie sposób przecenić wartości tego narzędzia i ilości czasu, jaki można zaoszczędzić, umiejętnie się nim posługując. Możliwość pełnego dekodowania informacji w warstwie protokołów aplikacyjnych i uzyskiwania zwięzłych informacji o tym, co się dzieje w tych aplikacjach, jest po prostu bezcenna.



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s
Frame	100.0	11132	100.0	8784511	759 k
Ethernet	100.0	11132	1.8	155848	13 k
Logical-Link Control	0.8	88	0.0	3344	289
Spanning Tree Protocol	0.8	88	0.0	3080	266
Internet Protocol Version 6	0.3	29	0.0	1160	100
User Datagram Protocol	0.2	25	0.0	200	17
Multicast Domain Name System	0.2	25	0.1	7808	674
Internet Control Message Protocol v6	0.0	4	0.0	128	11
Internet Protocol Version 4	97.8	10886	2.5	217720	18 k
User Datagram Protocol	4.7	526	0.0	4208	363
Simple Service Discovery Protocol	1.1	118	0.4	36760	3177
QUIC (Quick UDP Internet Connections)	0.2	24	0.1	7082	612
Network Time Protocol	0.0	2	0.0	96	8
NetBIOS Name Service	0.0	4	0.0	200	17
NetBIOS Datagram Service	0.0	3	0.0	662	57
SMB (Server Message Block Protocol)	0.0	3	0.0	416	35
SMB MailSlot Protocol	0.0	3	0.0	75	6
Microsoft Windows Browser Protocol	0.0	3	0.0	158	13
Multicast Domain Name System	0.4	40	0.1	10158	878
Domain Name System	2.6	292	0.3	22012	1902
Data	0.4	42	0.0	1080	93
Bootstrap Protocol	0.0	1	0.0	300	25
Transmission Control Protocol	93.0	10351	94.6	8307981	718 k
Secure Sockets Layer	27.8	3094	87.7	7702114	665 k
Malformed Packet	0.0	1	0.0	0	0
Hypertext Transfer Protocol	2.1	230	1.5	130470	11 k
Online Certificate Status Protocol	0.3	32	0.1	9632	832

Rysunek 2.7. Hierarchia protokołów w programie Wireshark

Ataki podsłuchowe

Większość nowoczesnych sieci jest zbudowanych z przełączników. Dzięki temu każde urządzenie źródłowe może wysyłać pakiety na określony port, do którego podłączone jest urządzenie docelowe. Dawniej sieci składały się z hubów. Każdy pakiet był rozsyłany do wszystkich portów huba, przez co wszystkie urządzenia mogły na podstawie adresu MAC „dowiedzieć się", kto jest dołączony do sieci. Huby nie miały w sobie żadnej inteligencji i działały jak zwykłe wzmacniacze.

Wszystko zmieniło się po wprowadzeniu przełączników. Przełącznik odczytuje docelowy adres MAC ramki z jej nagłówka utworzonego w warstwie danych. Wie również, do którego portu dołączone zostało urządzenie posiadające ten adres. Tę informację uzyskuje, obserwując ramki wysyłane z poszczególnych portów. Źródłowy adres MAC ramki wysyłanej przez dany interfejs jest taki sam jak adres dołączonego do niego urządzenia. Adresy przypisane do poszczególnych interfejsów przełącznik przechowuje w tablicy CAM (ang. *Content Addressable Memory*, pamięć adresowana zawartością). Nie sprawdza za każdym razem całej tablicy, tylko odczytuje z niej bezpośrednio informacje związane z danym adresem MAC, czyli oznaczenie interfejsu.

Dlaczego jest to takie ważne? Dlatego że czasem będziesz potrzebował informacji o systemie, do którego nie będziesz miał dostępu. Jeżeli zarządzasz siecią i masz dostęp do przełącznika, możesz go skonfigurować tak, aby kopiował ramki przesyłane pomiędzy co najmniej dwoma interfejsami i wysyłał je na jeszcze inny interfejs. Jest to tzw. powielanie ramek (nie mylić z kierowaniem). Ramki otrzymuje urządzenie docelowe, jak również inne urządzenie, które przechwytuje lub analizuje pakiety.

Jeżeli w zwykły sposób nie możesz uzyskać dostępu do pakietów, które Cię interesują, możesz przeprowadzić *atak podsłuchowy* polegający na podszyciu się pod inne urządzenie w celu przechwycenia wysyłanych i odbieranych przez nie pakietów. Atak można przeprowadzić na kilka opisanych niżej sposobów.

Uwaga etyczna

Ataki podsłuchowe są dokonywane przez hakerów, jednak nie powinieneś ich przeprowadzać w testowanej sieci (chyba że na tym właśnie polegają testy), ponieważ w ich wyniku mogą zostać ujawnione poufne dane.

Podsłuchiwanie protokołu ARP

Protokół ARP jest prosty. Gdy jakieś urządzenie potrzebuje skomunikować się z innym urządzeniem o znanym adresie IP, ale nieznanym adresie MAC, wysyła w sieć zapytanie `who-has` (kto ma dany adres?). Urządzenie o podanym w zapytaniu adresie IP odsyła odpowiedź `is-at` (mam adres) ze swoim adresem MAC. Urządzenie nadawcze uzyskuje w ten sposób adres odbiorcy i może wysyłać pakiety do właściwego miejsca.

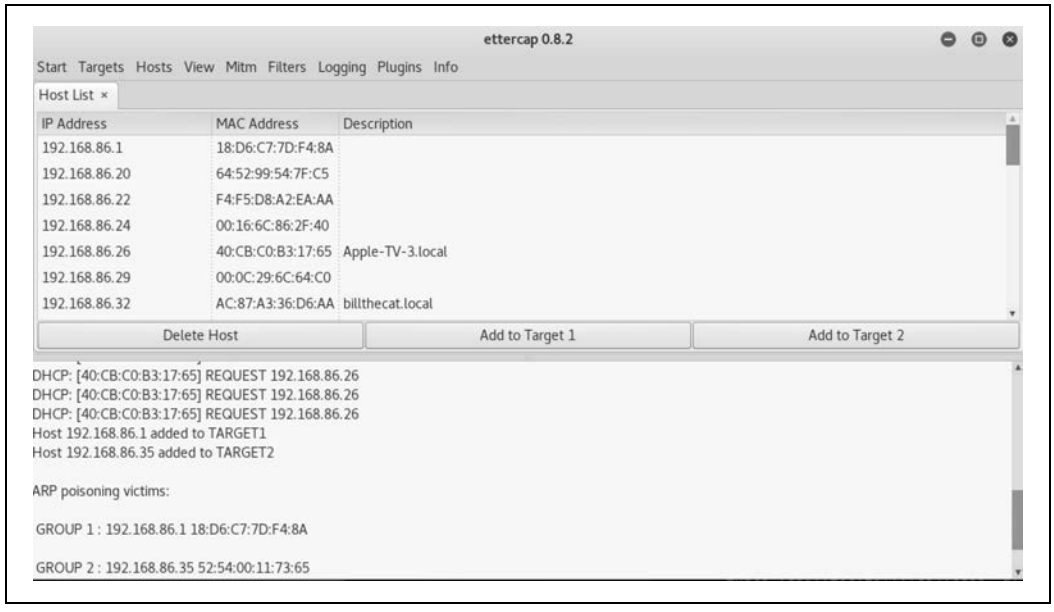
Aby komunikacja przebiegała sprawnie, system zapamiętuje uzyskaną tą drogą parę adresów. W rzeczywistości zapamiętuje wszystkie powiązane ze sobą adresy. W protokole ARP przyjęte zostało założenie, że urządzenie tylko wtedy może udzielić informacji, że posiada dany adres IP, gdy inne urządzenie o to zapyta. Jak się jednak okazuje, zasada ta nie jest przestrzegana. Jeżeli jakieś urządzenie wyśle odpowiedź `is-at` z adresem IP innego urządzenia i umieści w niej własny adres MAC, to może odbierać pakiety przeznaczone dla tego innego urządzenia i podsłuchiwać komunikację.

W ten sposób jednak przechwytywana jest komunikacja tylko w jedną stronę. Gdy wrogie urządzenie poda własny adres MAC jako powiązany z adresem IP innego urządzenia docelowego, będzie w stanie przechwytywać jedynie pakiety wysyłane do tego urządzenia. Aby podsłuchać komunikację w odwrotnym kierunku, urządzenie musi podsłuchać adres nadawcy. W tym celu może podszyć się pod bramę domyślną i przechwytywać pakiety przesyłane pomiędzy wybranym urządzeniem a internetem.

W opisany wyżej sposób można przechwytywać pakiety wysyłane do zadanego urządzenia. Trzeba też jednak wysyłać pakiety do nadawcy. Jeżeli się tego nie zrobi, połączenie szybko zostanie zamknięte, ponieważ nadawca nie będzie dostawał odpowiedzi na wysyłane pakiety. Dlatego urządzenie podsłuchujące musi wysyłać odpowiednie komunikaty do nadawcy.

Zgodnie z protokołem ARP powiązania pomiędzy adresami przedawniają się po pewnym czasie, a zatem jeżeli urządzenie na dłuższy czas przestanie wysyłać pakiety, straci możliwość wysyłania ich w ogóle. Aby temu zapobiec, musi wysyłać tzw. **samorzutne komunikaty ARP** (ang. *gratuitous ARP*), czyli odpowiedzi na nieistniejące zapytania. Urządzenia wysyłają takie komunikaty w kilku uzasadnionych, ale rzadkich przypadkach.

Do wykonania opisanego ataku można użyć różnych narzędzi, my zastosujemy program Ettercap. Program ten działa w dwóch trybach. Pierwszy to tryb tekstowy. W tym trybie program działa w oknie terminala, ale nie jest typowym programem terminalowym, ponieważ wykorzystuje tekstowy interfejs użytkownika. Drugi tryb to typowy tryb graficzny. Rysunek 2.8 przedstawia wygląd okna programu Ettercap po zaatakowaniu wybranych komputerów i podsłuchaniu protokołu ARP. W celu rozpoczęcia



Rysunek 2.8. Program Ettercap

ataku zostały podsłuchane adresy MAC wszystkich ramek przesłanych przez sieć, a następnie wybrane dwa adresy.

Powodem wybrania dwóch adresów była konieczność podszycia się pod dwie komunikujące się strony. W przypadku podszycia się tylko pod jedną z nich podsłuchiwana byłaby komunikacja odbywająca się tylko w jednym kierunku. W tym przykładzie celem było podsłuchanie pełnej komunikacji pomiędzy wybranym urządzeniem a internetem. Dlatego zostało wybrane jedno z urządzeń dołączonych do sieci oraz router. Aby podsłuchać pakiety przesyłane pomiędzy dwoma komputerami w tej samej sieci, należałoby oba wybrać jako cele ataku. Wtedy jedno byłoby w programie oznaczone jako *Target 1* (cel 1), a drugie jako *Target 2*. Listing 2.13 zawiera informacje o ramkach wysyłanych podczas podsłuchiwania protokołu ARP. Widoczne są dwie odpowiedzi na zapytania, zawierające adresy IP wybranych urządzeń. Listing przedstawia również część informacji uzyskanych za pomocą polecenia `ifconfig`, którego użyłem w swoim systemie. Jak widać, jednym z podsłuchanych adresów MAC jest adres interfejsu systemu, za pomocą którego przeprowadziłem atak.

Listing 2.13. Informacje wyświetlane przez program tcpdump podczas podsłuchiwania protokołu ARP

```
17:06:46.690545 ARP, Reply rosebud.lan is-at 00:0c:29:94:ce:06 (oui Unknown), length 28
17:06:46.690741 ARP, Reply testwifi.here is-at 00:0c:29:94:ce:06 (oui Unknown), length 28
17:06:46.786532 ARP, Request who-has localhost.lan tell savagewood.lan, length 46
^C
43 packets captured
43 packets received by filter
0 packets dropped by kernel
root@kali:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.86.227  netmask 255.255.255.0  broadcast 192.168.86.255
        inet6 fe80::20c:29ff:fe94:ce06  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:94:ce:06  txqueuelen 1000  (Ethernet)
```

Po zakończeniu ataku możesz zacząć przechwytywać przesyłane pakiety za pomocą programu `tcpdump` lub Wireshark. Pamiętaj, że przeprowadzenie tego rodzaju ataku jest możliwe tylko w lokalnej sieci, ponieważ wykorzystywane są w nim adresy MAC stosowane jedynie w protokołach warstwy danych. Adresy te nie są używane w protokołach wyższych warstw (nie są przekazywane pomiędzy różnymi sieciami).

Program Ettercap umożliwia również przeprowadzanie innych ataków na protokoły stosowane w warstwie danych, na przykład podsłuchiwanie DHCP i przekierowanie ICMP. Oba ataki można wykorzystać do przechwycenia pakietów przesyłanych w obrębie lokalnej sieci.

Podsłuchiwanie systemu DNS

Pakiety wysyłane poza lokalną sieć przechwytuje się między innymi po to, aby móc przeprowadzić atak polegający na podszyciu się pod system DNS. Celem ataku jest przechwycenie zapytań, które wybrany komputer wysyła do serwera DNS, aby uzyskać adres IP innego komputera o zadanej nazwie. W wyniku ataku na przechwycone zapytanie wysyłana jest odpowiedź zawierająca adres IP jeszcze innego, kontrolowanego przez hakera komputera. Tego rodzaju atak jest niekiedy nazywany **zatruwaniem bufora DNS**, ponieważ polega on na przejmowaniu funkcji serwerów znajdujących się w pobliżu urządzenia będącego celem ataku. Takim serwerem może być komputer buforujący przez uzgodniony czas adresy w imieniu legalnych serwerów DNS.

Kiedy przejmie się funkcję serwera buforującego, można zmodyfikować pamięć podręczną tak, aby zapytania wysyłane przez komputer będący celem ataku kierować do innego, kontrolowanego systemu. Ponadto w odpowiedziach na zapytania można umieszczać nieistniejące adresy IP i w ten sposób utrudniać komunikację wszystkim komputerom korzystającym z serwera buforującego. Atak ten ma tę „zaletę", że można go przeprowadzić spoza lokalnej sieci. Wymagane jest jednak przechwycenie komunikacji z zewnętrznym serwerem DNS.

Prostszy w użyciu jest program `dnsspoof`, z tym że jego zastosowanie ogranicza się do sieci lokalnej. Urządzenie po wysłaniu zapytania do serwera DNS oczekuje od niego odpowiedzi. Zapytanie zawiera identyfikator, który z założenia ma uniemożliwiać hakerom wysyłanie fałszywych odpowiedzi. Jednak gdy haker przechwyci zapytanie, wtedy może w swojej odpowiedzi umieścić identyfikator wraz z adresem IP własnego komputera. Program `dnsspoof` napisał Dug Song wiele lat temu, gdy sieci zbudowane z przełączników były rzadkością. Jeżeli w sieci wykorzystywane są przełączniki, to w celu przechwycenia zapytań wysyłanych do serwera DNS trzeba wykonać dodatkową operację.

Program `dnsspoof` uruchamia się bardzo łatwo, jednak trudniejsze jest przygotowanie go do uruchomienia. Potrzebny jest plik, w którym poszczególne wiersze zawierają adresy IP komputerów i ich rzekome nazwy oddzielone spacjami. Po przygotowaniu takiego pliku można uruchomić program `dnsspoof`, jak w listingu 2.14.

Listing 2.14. Przykład użycia programu dnsspoof

```
root@kali:~# dnsspoof -i eth0 -f myhosts udp dst port 53
dnsspoof: listening on eth0 [udp dst port 53]
192.168.86.227.37972 > 192.168.86.1.53:  10986+ A? www.bogusserver.com
192.168.86.227.49273 > 192.168.86.1.53:  28879+ A? www.bogusserver.com
192.168.86.227.48253 > 192.168.86.1.53:  53068+ A? www.bogusserver.com
192.168.86.227.49218 > 192.168.86.1.53:  45265+ A? www.bogusserver.com
```

Jak widać, na końcu polecenia znajduje się filtr BPF, dzięki któremu przechwytywane są tylko wybrane zapytania. Program `dnsspoof` uruchomiony bez filtra przechwytuje pakiety UDP wysyłane na port numer 53 przez komputery inne niż lokalny komputer. Użyty tutaj filtr umożliwia wykonanie testu na lokalnym systemie. Jak widać, odbierane zapytania są znakowane. Program `dnsspoof` wyświetla podobne informacje jak `tcpdump`.

Być może zastanawiasz się, dlaczego trzeba wykonywać dodatkową operację polegającą na użyciu programu `dnsspoof`, skoro można bez przeszkód korzystać z programów Ettercap lub `arpspoof` (jest to inny program do podsłuchiwania protokołu ARP, umieszczony w tym samem zestawie narzędzi co `dnsspoof`, również napisany przez Duga Songa). Otóż za pomocą modułu `dnsspoof` można osiągnąć to, czego nie da się uzyskać, podsłuchując protokół ARP, tj. można przekierować zapytania na adres IP komputera udającego legalny system. W efekcie można utworzyć na przykład wrogi serwer WWW, który wygląda tak samo jak legalny, lecz zawiera złośliwy kod infekujący komputery lub wyłudzający dane. Jest to częsty, choć nie jedyny powód podszywania się pod serwer DNS.

Podsumowanie

Zazwyczaj ataki na systemy dokonywane są poprzez sieć. Choć nie w każdym ataku wykorzystywane są protokoły sieciowe, to mimo wszystko warto poświęcić czas na poznanie protokołów stosowanych w różnych warstwach stosu sieciowego.

Poniżej wymienione są najważniejsze zagadnienia opisane w tym rozdziale:

- Testowanie bezpieczeństwa danych polega na wykrywaniu zagrożeń dla ich poufności, spójności i dostępności.
- Stos protokołów sieciowych oparty na modelu OSI składa się z następujących warstw: fizycznej, danych, sieciowej, transportowej, sesji, prezentacji i aplikacji.
- Testy obciążeniowe ujawniają przynajmniej zagrożenie dla dostępności danych.
- Szyfrowanie danych utrudnia podsłuchanie komunikacji, jednak słabe szyfrowanie może narażać poufność danych.
- W wyniku ataku podsłuchowego mogą zostać przechwycone pakiety pochodzące z zewnętrznych źródeł.
- Za pomocą programów przechwytujących pakiety, takich jak `tcpdump` lub Wireshark, można sprawdzać, co się dzieje z aplikacjami.
- System Kali oferuje narzędzia przydatne do testowania bezpieczeństwa sieci.

Przydatne materiały

- Strona Duga Songa poświęcona pakietowi `dsniff` (*https://monkey.org/~dugsong/dsniff*).
- Film Rica Messiera *TCP/IP*, Infinite Skills, 2013 (*http://bit.ly/tcp-ip-video*).
- Craig Hunt, *TCP/IP Network Administration*, wyd. 3, O'Reilly, 2010 (*http://bit.ly/tcp-ip-network-admin-3e*).

Skorowidz

C

D

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

Ś

T

U

V

W

X

Y

Z

O'REILLY®

Kali Linux — jak bezpieczny jest Twój system?

Kali Linux jest specjalistyczną dystrybucją systemu Linux, którą przeznaczono do celów związanych z bezpieczeństwem IT. Udostępnia kilkaset narzędzi między innymi do testowania zabezpieczeń, tworzenia eksploitów, dekodowania aplikacji lub po prostu śledzenia nadużyć i incydentów bezpieczeństwa. Sporo z tych narzędzi pozwala na stosowanie zaawansowanych praktyk, takich jak testy penetracyjne czy techniki inżynierii wstecznej. Szerokie możliwości Kali mogą jednak przytłaczać nawet biegłych specjalistów. Tymczasem zapewnienie bezpieczeństwa IT wymaga wiedzy i umiejętności wyboru programu najwłaściwszego do wykonania potrzebnego testu.

Ta książka jest praktycznym przewodnikiem po systemie Kali Linux, zawierającym szczegółowe informacje o jego możliwościach. Najwięcej uwagi poświęcono udostępnianym w nim narzędziom, które nie są zbyt popularne w innych dystrybucjach Linuksa. Poza podstawami budowy i działania systemu Kali Linux opisano tu metody testowania sieci, aplikacji WWW, sieci bezprzewodowych, siły haseł itp. Pokazano też różne techniki rozszerzania systemu o nowe narzędzia i tworzenia ich zestawów, w pełni odpowiadających specyficznym potrzebom. Równolegle w książce omówiono zagadnienia bezpieczeństwa systemów IT, w tym ich podatności, które wskazują na potrzebę przeprowadzania odpowiednich testów.

Ric Messier — od początku lat 80. zeszłego stulecia interesuje się zagadnieniami bezpieczeństwa. Od ponad ćwierćwiecza zajmuje się systemami Unix i Linux. Uzyskał takie certyfikaty jak GCIH, GSEC, CEH, CISSP. Jest autorem publikacji, instruktorem, wykładowcą, niepoprawnym kolekcjonerem certyfikatów branżowych i specjalistą do spraw bezpieczeństwa z kilkudziesięcioletnim doświadczeniem.

W tej książce między innymi:

- podstawy Kali Linux i testowania bezpieczeństwa
- techniki rekonesansu sieciowego i wyszukiwania słabych punktów
- eksploity i platforma Metasploit
- sieci bezprzewodowe: skanowanie, wstawianie ramek danych, łamanie haseł
- techniki zaawansowane
- raportowanie i wnioski z przeprowadzonych testów

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

ISBN 978-83-283-5426-5
9 788328 354265

INFORMATYKA W NAJLEPSZYM WYDANIU

Cena: 67,00 zł