Firewall-e i bezpieczeństwo w sieci. Vademecum profesjonalisty

Autorzy: William R. Chestwick, Steven M. Bellovin, Aviel D. Ru
Tłumaczenie: Arkadiusz Romanek, Witold Zioło
ISBN: 83-7361-209-2
Tytuł oryginału: Firewalls and Internet Security Second Edition
Format: B5, stron: 444

Książka „Firewalle i bezpieczeństwo w sieci. Vademecum profesjonalisty" to kompletne kompendium wiedzy na temat zabezpieczania sieci komputerowych. Autorzy skoncentrowali się na omówieniu jednego z najważniejszych elementów, stosowanych z myślą o bezpieczeństwie: firewalla, przekazując wiele cennych wskazówek na temat jego konfiguracji. Znajdziesz tu jednak nie tylko informację o zaporach sieciowych, ale także spojrzenie na bezpieczeństwo sieci z szerszej perspektywy. Poznasz najważniejsze techniki ataku i programy stosowane przez włamywaczy; prześledzisz kolejne kroki, które podejmowali hakerzy, by uzyskać dostęp do chronionych danych. A co najważniejsze, nauczysz się lokalizować zagrożenia i zapobiegać im.

- Dogłębna analiza związanych z bezpieczeństwem aspektów protokołu TCP/IP
- Projektowanie i instalacja firewalli - krok po kroku
- Narzędzia monitorujące działanie firewalli
- Darmowe narzędzia zabezpieczające
- Z życia wzięte przykłady włamań i analiza użytych technik
- Prawne aspekty zabezpieczania sieci komputerowych
- Techniki kryptograficzne

Jeśli jesteś odpowiedzialny za bezpieczeństwo sieci, nie obędziesz się bez tej książki. Jej przystępny język sprawi, że z zaciekawieniem przeczytają ją także wszyscy zainteresowani najnowszymi technikami hakerskimi, a także osoby, które chcą pogłębić swoją wiedzę na temat Internetu.

Bezpieczeństwo w Internecie jest ostatnio modnym tematem; stało się tematem filmów, książek i dreszczowców. Zadbaj o to, aby scenariusze wielu „mrożących krew w żyłach" opowieści nie stały się Twoim udziałem.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

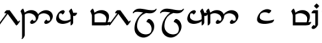
Spis treści

Rozdział 5.

Klasy ataków

Omówiliśmy dotychczas kilka technik atakowania systemów. Wiele z nich ma podobne cechy. Techniki te warto usystematyzować, ponieważ wzorce jakie powstaną dzięki systematyzacji, pokażą, gdzie konieczne jest zacieśnienie ochrony.

5.1. Kradzieże haseł



„(Powiedz przyjacielu i wejdź)

Co to ma znaczyć: Powiedz przyjacielu i wejdź? — spytał Merry.

Sens dość jasny — rzekł Gimilo. — Jeżeli jesteś przyjacielem, powiedz hasło, a drzwi się otworzą i będziesz mógł wejść.

…

Ale ty, Gandalfie, chyba znasz hasło? — spytał zdumiony Boromir.

Nie! — odparł Czarodziej. — … jeszcze nie znam hasła … ale wkrótce je poznamy".

Władca pierścieni
J.R.R. Tolkien

Najprostsze wejście do komputera prowadzi zwykle przez jego drzwi wejściowe, czyli przez polecenie *login*. W przypadku praktycznie wszystkich systemów, skuteczne logowanie bazuje na podaniu, w określonej liczbie prób, prawidłowego hasła.

W historii typowych programów logowania (także nie przeznaczonych dla systemów uniksowych) zaobserwować serię nasilających się ataków i wzmożone wysiłki wkładane w obronę przed nimi, istny wyścig zbrojeń. Znane są nam wczesne systemy operacyjne, które przechowywały w plikach dyskowych hasła w czystej, tekstowej postaci. Bezpieczeństwo jednego z tych systemów bazowało na tajności nazwy pliku przechowującego hasło, zatem każdy, kto znał tę nazwę mógł odczytać hasło. Bezpieczeństwo systemu było „chronione" w ten sposób, że systemowe polecenie przedstawiające zawartość katalogu pomijało nazwę tego pliku (wywołana funkcja systemowa podawała tę nazwę).

To podejście do bezpieczeństwa opierało się na *zabezpieczaniu przez ukrywanie* (ang. *security by obscurity*). Ukrywanie nie jest wcale złą opcją, ale przez tego typu działania straciło dobrą sławę. W końcu klucz kryptograficzny jest też jakimś sposobem ukrywania, tylko że dobrze przemyślanym. Błędem tu popełnionym była słabość ukrywania oraz brak innych poziomów obrony.

Błędy w systemach stwarzają intrygujące możliwości włamań do systemu, ale nie jest to najłatwiejszy ze sposobów ataku na system. Ten honor należy raczej oddać przyziemnemu atakowi na hasła użytkowników. Znaczny procent włamań do systemów następuje z powodu uchybień w całym systemie haseł.

Mówimy o „systemie haseł", gdyż jest klika przyczyn takich uchybień. Jednak najpowszechniejszym uchybieniem jest stosowanie przez użytkowników bardzo słabych haseł. Powtarzane badania pokazały, że próba odgadnięcia hasła zwykle kończy się powodzeniem, czego przykłady można znaleźć w pracach Kleina [1990] oraz Morrisa i Thompsona [1979]. Nie mówimy tu, że *każdy* używa złych haseł, ale należy zdawać sobie sprawę, że napastnikowi wystarczy tylko jedno złe hasło.

Ataki zgadywania haseł przyjmują dwie podstawowe formy. Pierwsza polega na usiłowaniu zalogowania się za pomocą znanej lub założonej nazwy użytkownika oraz zgadywanego hasła. Ta metoda kończy się sukcesem niezwykle często, ponieważ w wielu miejscach często stosuje się parę „nazwa użytkownika-hasło", takich jak *field*-`service`, *guest*-`guest`, itp. Nierzadko pary te kopiowane są wprost z podręczników systemowych! Pierwsza próba może się nie powieść, może nie powieść się nawet dziesiąta, ale mimo wszystko, któraś okaże się trafiona, a gdy napastnik znajdzie się w środku, będzie już poza główną linią obrony. Niestety, niewiele systemów operacyjnych potrafi odpierać ataki pochodzące z ich wnętrza.

Taka metoda włamywania w ogóle nie powinna być możliwa! Użytkownikom nie powinno się pozostawiać nieograniczonej liczby prób logowania się z niepoprawnym hasłem, wszystkie niepowodzenia logowania powinny być odnotowywane, użytkownicy powinni być powiadamiani o zakończonych niepowodzeniem próbach dostania się do ich konta, itd. Żadne z zaleceń, o których mowa, nie jest nowe, tyle że takie środki bezpieczeństwa podejmuje się rzadko, a właściwie bardzo rzadko. Wiele z najczęściej popełnianych błędów wymienionych zostało w pracy Gramppa i Morrisa [1984], ale niewielu projektantów systemów wzięło pod uwagę ich rady. Co gorsza, większość logowań w systemach uniksowych odbywa się za pomocą poleceń *login* oraz *su*. Inne programy wykorzystujące hasła, takie jak *ftpd*, *rexecd*, różne programy blokujące ekrany itp., w większości systemów nie odnotowują niepowodzeń logowania. Co więcej, nawet w systemach poprawnie rejestrujących takie zdarzenia, administratorzy nie przeglądają dzienników zdarzeń regularnie. Oczywiście, dzienniki zdarzeń zawierające nazwy użytkowników, którzy nie zalogowali się poprawnie będą niezmiennie zawierały jakieś hasła.

Drugim sposobem zdobywania haseł przez napastników jest odgadywanie haseł znajdujących się w skradzionych plikach przechowujących hasła (w systemach uniksowych w pliku */etc/passwd*). Pliki te kradzione są z systemów, do których już się włamano. W takim przypadku napastnicy będą próbowali zastosować zdobyte hasła w innych komputerach

(użytkownicy często posługują się na różnych komputerach tymi samymi hasłami). Hasła mogą również zostać zdobyte z systemów, które jeszcze nie zostały spenetrowane. Ataki tego rodzaju nazywane są *atakami słownikowymi* (ang. *dictionary attacks*) i zwykle kończą się pełnym powodzeniem. Nie ma wątpliwości, że bezpieczeństwo komputera *zostanie* naruszone, jeżeli plik z hasłami dostał się w ręce wroga. Klein [1990] informuje, że 25% haseł jest kradzionych. Jeżeli ta liczba jest prawdziwa, w przypadku komputera z zaledwie szesnastoma kontami użytkowników, istnieje 99% prawdopodobieństwo, że co najmniej jedno z tych haseł zostanie osłabione.

Również szyfrowanie na nic się nie zda, gdy klucze zostaną odtworzone z haseł użytkowników. Eksperymenty z Kerberosem [Wu, 1999] pokazały to dobitnie.

Trzecim sposobem jest zdobycie hasła przez podsłuchanie prawidłowej sesji terminalowej. Przy wykorzystaniu tej metody, niezależnie od tego, jak dobre było hasło, konto i prawdopodobnie cały system staną się zagrożone.

Jak długie powinno być hasło?

Panuje powszechna zgoda, że dotychczasowy wymóg systemów uniksowych stosowania haseł o długości co najmniej ośmiu znaków nie jest już wystarczający [Feldmeier i Karn, 1990; Leong i Tham, 1991]. Wobec tego, jak długie powinno być hasło?

Problem z algorytmem matematycznego mieszania haseł stosowanym w systemie Unix polega na tym, że jako klucza szyfrującego używa on wprost siedmiu znaczących bitów każdego podanego znaku hasła. Z powodu zastosowanego algorytmu [DES, NBS, 1977] używane są tylko klucze o długości 56 bitów. Limit ośmiu znaków nie został zatem założony, tylko wynikał wprost z długości użytego klucza. To jednak powoduje dalsze wątpliwości.

128 możliwych kombinacji wywodzących się z siedmiu bitów nie jest jednakowo prawdopodobnych. Ludzie nie tylko unikają stosowania w hasłach znaków sterujących, ale większość z nich nie używa innych znaków niż litery. Na dodatek, większość ludzi tworzy hasła składające się wyłącznie z małych liter.

Prawdziwą wartość haseł jako kluczy można poznać wykorzystując do tego celu teorię informacji [Shannon, 1949]. W przypadku zwykłego tekstu języka angielskiego składającego się z ośmiu znaków, zawartość informacji wynosi około 2,3 bita na jedną literę, może nawet mniej [Shannon, 1948, 1951]. W ten sposób, w hasłach w postaci słów języka angielskiego, uzyskujemy faktyczną długość klucza wynoszącą 19, a nie 56 bitów.

Niektóre osoby na hasła wybierają imiona (własne, małżonka, dzieci, itp.). To tylko pogarsza sprawę, gdyż niektóre imiona są bardzo powszechne. Eksperymenty przeprowadzane przez AT&T na elektronicznej książce telefonicznej pokazały, że imię ma tylko około 7,8 bita informacji w całym imieniu i nazwisku. To naprawdę bardzo zły wybór.

Dłuższe wyrażenia języka angielskiego mają mniejszą zawartość informacji przypadającą na jedną literę — od 1,2 do 1,5 bita. Zatem hasła składające się z 16 bajtów, zawierające wyrażenia języka angielskiego, nie są tak silne jakby się mogło wydawać, gdyż zawierają tylko od 19 do 24 bitów informacji. Sytuacja ta poprawi się do 38 bitów, gdy użytkownik na hasło wybierze niezależne od siebie słowa. Jednak wybranie jako hasła kombinacji imion nie będzie zbyt pomocne.

Przy tak powszechnym dostępie do haseł, nie powinno się ich używać w ogóle lub przynajmniej powinny być one kryptograficznie zabezpieczane przed atakami słownikowymi.

W tym miejscu można już wysunąć kilka wniosków. Po pierwsze, bardzo ważne jest nauczenie użytkowników tworzenia dobrych haseł. Niestety, mimo upływu lat od wydania przez Morrisa i Thompsona [1979] pracy na ten temat, zwyczaje użytkowników nie zmieniły się. Nie pomogło też nakładanie ściślejszych ograniczeń na hasła, które można wykorzystać, mimo że podjęto wiele takich prób [Spafford, 1992; Bishop, 1992]. Inni próbowali poprawić bezpieczeństwo haseł, sprawdzając już używane hasła [Muffett, 1992]. Jednak przekora cechująca człowieka zawsze jest silniejsza, a hakerom wystarczy jedna wygrana.

Ludzie używają kiepskich haseł, co taka jest ludzka natura. Podejmowano wiele prób zmuszenia ludzi do używania haseł trudnych do odgadnięcia [Brand i Makey, 1985], ale nie przyniosły one skutku. Żeby włamać się do komputera wystarczy włamać się tylko na jedno konto, a napastnicy wyposażeni w małe słowniki osiągają sukces w ponad 20% przypadków [Klein, 1990]. Duże słowniki mogą osiągać wielkość dziesiątków megabajtów. Słowniki te składają się ze słów oraz rdzeni wyrazów większości języków pisanych. Mogą zawierać też informacje osobiste, takie jak: numer pokoju, numer telefonu, hobby, imiona i nazwiska ulubionych autorów, itp. W niektórych systemach wiele z tych informacji zawartych jest w plikach haseł, inne z radością dostarczy każdemu, kto o nie poprosi, program *finger*.

W przypadku wielu ataków sieciowych celem nie jest wcale bezpośrednie włamanie się do systemu — co jest zwykle trudniejsze, niż powszechnie się sądzi — ale wykradnięcie pliku haseł. Do usług, dzięki którym udało się zdobyć pliki haseł, należą: FTP, TFTP, system pocztowy, NIS, *rsh*, *finger*, *uucp*, X11 i wiele innych. Innymi słowy, napastnik łatwo osiągnie swój zamiar, gdy administrator postępuje niedbale lub nie miał szczęścia przy wyborze systemu dla komputera. Środków obrony przed atakami to w dużej mierze wielka dbałość o system oraz ostrożne podejście do oprogramowania.

Jeżeli nie można wymusić na użytkownikach używania dobrych haseł, istotne staje się przechowywanie pliku haseł z dala od wrogów. Oznacza to, że należy:

- starannie konfigurować funkcje bezpieczeństwa usług, takich jak NIS firmy Sun,
- ograniczać korzystanie z plików przez *tftpd*,
- unikać umieszczania właściwego pliku */etc/passwd* w obszarze osiągalnym przez anonimowy dostęp przez FTP.

Niektóre systemy uniksowe umożliwiają ukrywanie zaszyfrowanych haseł nawet przed uprawnionymi użytkownikami. Jeżeli system wyposażony jest w tę funkcję, nazywaną plikiem *przesłoniętym* (ang. *shadow*) lub *dodatkowym* (ang. *adjunct*), zdecydowanie polecamy skorzystanie z niej. Wiele systemów operacyjnych mądrze szyfruje i ukrywa swoje pliki haseł.

Idealnym wyjściem byłoby całkowite uwolnienie się od haseł. Najlepsze jest uwierzytelnianie przy użyciu żetonów, a przynajmniej używanie systemów jednorazowych haseł, takich jak *One-Time Password (OTP)* [Haller, 1994; Haller i Metz, 1996]. Oczywiście, w takim przypadku również należy wystrzegać się haseł łatwych do odgadnięcia.

5.2. Socjotechnika

„Musimy uruchomić system."

…

Strażnik odchrząknął i rzucił tęsknym wzrokiem na swoją książkę.
„Uruchamianie systemu nie należy do moich obowiązków. Przyjdźcie jutro."

„Jeżeli nie uruchomimy teraz systemu, zrobi się gorąco. Przegrzeje się.
Muy caliente i dużo pieniędzy."

Okrągła i pulchna twarz strażnika zmarszczyła się ze strachu, ale poruszył tylko ramionami. „Ja tego nie potrafię zrobić, co miałbym zrobić?"

„Wiem, że masz klucze. Wpuść nas, my to zrobimy."

Strażnik mrugnął urażony. „Tego nie zrobię" oświadczył. „To jest zabronione".

…

„Czy widziałeś kiedyś rozbity komputer?" nalegał. „To wygląda strasznie.
Pełno go na całej podłodze!"

Herbata z Czarnym Smokiem
R.A. MacAvoy

Często najstarsze sposoby sprawdzają się najlepiej. Hasła można znaleźć na kartkach przyczepionych do terminala lub zapisane w dokumentacji leżącej obok klawiatury (do ich uzyskania potrzebny jest jednak fizyczny dostęp, który nie jest w naszej książce głównym przedmiotem zainteresowań). Podejście wykorzystujące socjotechnikę zwykle opiera się na użyciu telefonu i wykazaniu się odrobiną tupetu. Oto prawdziwy przykład z AT&T:

„Mówi Ken Thompson. Ktoś dzwonił do mnie w sprawie problemu z poleceniem *ls*.
Ta osoba chciała, żebym się tym zajął."

„Taaak? Dobrze. Co mam zrobić?"

„Wystarczy zmienić hasło mojego logowania w twoim komputerze
— nie używałem go od jakiegoś czasu."

„Dobrze, nie ma problemu."

Istnieje jeszcze kilka innych sposobów, takich jak choćby fałszowanie adresu nadawcy poczty. Poradnik CERT [CA-91-04, 18 kwietnia 1991] ostrzega przed wiadomościami pocztowymi, jakoby pochodzącymi od administratora systemu, w których użytkownicy proszeni są o uruchomienie „programu testowego" wymagającego podania hasła.

Napastnicy również wysyłają takie wiadomości:

```
From: smb@research.att.com
To: admin@research.att.com
Subject: Gość
```

```
W przyszłym tygodniu będziemy mieli gościa. Mógłbyś poprosić administratora systemu
o założenie dla niego konta w systemie? Podaję Ci wiersz hasła, użyj tego samego
zaszyfrowanego hasła.
pxf:5bHD/k5k2mTTs:2403:147:Pat:/home/pat:/bin/sh
```

Należy zauważyć, że postąpienie według tej procedury byłoby błędem, nawet gdyby sama wiadomość była prawdziwa. Jeżeli faktycznie ma pojawić się gość, nie powinien on używać takiego samego hasła w tym komputerze, jakiego używa na własnym. Jest to dość wygodny sposób na utworzenia nowego konta, ale tylko wtedy, gdy można zaufać gościowi, że zmieni swoje hasło na inne, zanim ktoś zrobi z niego użytek. Z drugiej strony, unika się w ten sposób wysyłania haseł pocztą w czystej, tekstowej postaci. Jak sobie pościelisz, tak się wyśpisz.

Pewnych czynności w ogóle nie powinno się wykonywać bez stosowania silnego uwierzytelnienia. Trzeba dobrze *wiedzieć*, kto prosi o daną rzecz. Uwierzytelnianie oczywiście nie musi być formalne. Jeden z nas „podpisał" ostatnio ważną wiadomość pocztową, cytując temat dyskusji prowadzonej przez nas podczas ostatniego obiadu. W wielu, ale nie we wszystkich, sytuacjach wystarczy nieformalne, „trójstopniowe potwierdzenie" w postaci zapytania i odpowiedzi następujących po właściwej prośbie. Sposób ten nie jest jednak niezawodny. Nawet konto uprzywilejowanego użytkownika może zostać zaatakowane.

Do bardziej wiarygodnego uwierzytelniania zaleca się korzystanie z kryptograficznych systemów pocztowych, które zostaną opisane w rozdziale 18. Pamiętać tylko należy, że żaden system kryptograficzny nie jest bardziej bezpieczny niż komputer, na którym działa. Sama wiadomość może być chroniona szyfrem, którego nie mogłaby złamać nawet NSA[1], ale gdyby tylko hakerowi udało się zastawić pułapkę w procedurze pytającej się o hasło, poczta przestałaby być bezpieczna i autentyczna.

Czasem ludzie mający dobre intencje, ale zbyt małą wiedzę, przyczyniają się do rozpowszechniania ataków wykorzystujących socjotechnikę. Często przecież zdarza się otrzymać od przyjaciela wiadomość pocztową informującą na przykład, że plik *sulfnbk.exe* jest wirusem i konieczne jest jego usunięcie, a na dodatek trzeba NATYCHMIAST ostrzec o tym fakcie inne osoby. To zwyczajnie bezsensowny żart, który na dodatek, gdyby odbiorca posłuchał tej porady, może okazać się niebezpieczny dla komputera. Niestety, wielu ludzi daje się nabrać na taki żart, gdyż o niebezpieczeństwie ostrzega zaufany przyjaciel czy kolega.

Osobom zaciekawionym zagadnieniem praktycznego wykorzystania socjotechniki polecamy relację byłego napastnika, Mitinicka [2002].

5.3. Błędy i tylne wejścia

Jednym ze sposobów rozprzestrzeniania się Robaka internetowego [ang. *Internet Worm*, Spafford, 1989a, 1989b; Eichin i Rochlis, 1989; Rochlis i Eichin, 1989] było wysyłanie nowego kodu programu demonowi *finger*. Naturalnie, demon nie był przygotowany na otrzymanie jakiegokolwiek kodu i w protokole nie zawarto w ogóle możliwości przyjęcia

[1] Agencja Bezpieczeństwa Narodowego Stanów Zjednoczonych — *przyp. tłum*.

kodu. Jednak program wywoływał funkcję `gets`, która nie określała maksymalnej wielkości bufora. Robak wypełniał swoim kodem cały bufor odczytu i jeszcze większy obszar poza nim do momentu, gdy nadpisał adres powrotny funkcji `gets` w module jej stosu. Kiedy wykonanie funkcji zostało zakończone, nastąpił skok do adresu umiejscowionego w buforze, w którym znajdował się kod napastnika. Konsekwencje są już znane.

Technika przepełniania bufora znana jest pod nazwą *stack-smashing* i jest najczęściej stosowanym sposobem łamania programów. Stworzenie takiego kodu wymaga trochę wysiłku, gdyż wpisywane znaki są kodem maszynowym atakowanego komputera, ale i tak wielu ludzi to robi. Historia komputerów oraz literatura zawierają liczne przykłady na to, jak uniknąć lub udaremnić przepełnienie bufora. W wielu językach komputerowych nie jest to w ogóle możliwe. Niektóre komputery (jak starsze urządzenia firmy Burroughs) nie wykonałyby kodu znajdującego się w obszarze stosu. Ponadto, wiele kompilatorów języka C oraz jego bibliotek używa różnych metod udaremniania lub wykrywania prób zrujnowania stosu.

Ta konkretna luka i jej proste odpowiedniki dawno już zostały naprawione przez większość producentów systemów, ale pozostaje jednak inna zasadnicza kwestia. Napisanie *poprawnego* programu wydaje się być problemem przekraczającym możliwości informatyki. Błędy się mnożą.

Błędem programu będziemy nazywać w programie to, co nie spełnia jego wymogów (czy wymogi te są poprawne, czy nie będziemy rozważać później). Błędy są przez to szczególnie trudne do zamodelowania, gdyż z definicji nie wiadomo, które z założeń (jeśli w ogóle któreś) zawiedzie.

„Pomarańczowa księga" [Brand, 1985, ramka na stronie 142] zawiera szereg kryteriów stworzonych przez Departament Obrony w celu określenia poziomu bezpieczeństwa systemu. W przypadku opisanego tu robaka nie pomogłyby żadne najbardziej strukturalne zabezpieczenia systemu wywodzące się z tej księgi. W najlepszym razie system o najwyższym poziomie bezpieczeństwa ograniczyłby naruszenie ochrony do pojedynczego poziomu bezpieczeństwa. Robak ten był faktycznie atakiem blokady usług, co nie jest wcale takie ważne, gdy komputer o wielopoziomowym systemie bezpieczeństwa zostaje zniszczony przez jawny lub nawet ściśle tajny proces. W każdym z tych przypadków system staje się bezużyteczny.

„Pomarańczowa księga" próbuje poradzić sobie z takimi sprawami przez koncentrowanie się na wymaganiach procesu i gwarancji bezpieczeństwa dla systemów o wyższych klasach. Wymagania na klasę B3 zawierają w punkcie 3.3.3.1.1 następujące zalecenie:

TCB (wiarygodna platforma komputerowa) powinna być zaprojektowana i skonstruowana tak, by używać kompletnego, koncepcyjnie prostego mechanizmu ochrony o ściśle zdefiniowanej semantyce. Ten mechanizm powinien odgrywać główną rolę we wdrażaniu wewnętrznych struktur TCB oraz systemu. TCB powinien wykorzystywać w znacznym stopniu warstwowość, abstrakcje oraz ukrywanie danych. Znaczący wysiłek projektowy powinien zostać skierowany na minimalizację złożoności TCB oraz na wyłączenie z TCB modułów, które nie są istotne dla bezpieczeństwa.

Standardy bezpieczeństwa informatycznego

Jaki komputer jest bezpieczny? Skąd wiadomo, czy się takim dysponuje? Albo inaczej, skąd wiadomo, że ktoś sprzedaje takie komputery?

Departament Obrony Stanów Zjednoczonych zajął się tym zagadnieniem we wczesnych latach osiemdziesiątych ubiegłego stulecia tworząc tak zwaną *Tęczową serię* (ang. *Rainbow Series*). Tęczowa seria była zbiorem broszur, które różniły się kolorem okładek i w których podejmowano różne tematy. Najsłynniejsza stała się *Pomarańczowa księga* [ang. *Orange book*, Brand, 1985] opisująca zbiór poziomów bezpieczeństwa oznaczony literami od D (najmniej bezpieczny) do A1. W kolejnych poziomach zwiększały się zarówno funkcje bezpieczeństwa, jak i pewność, że zostały one poprawnie zaimplementowane. W efekcie definicja „bezpieczeństwa" sprowadzała się do spełnienia wymagań modelu bezpieczeństwa odpowiadającego ściśle systemowi klasyfikacji Departamentu Obrony.

Był tylko jeden problem, idea bezpieczeństwa Departamentu Obrony nie pasowała do oczekiwań innych ludzi. Co gorsza, tematyka „Pomarańczowej księgi" została oparta na niejasnym założeniu, że dotyczy ona drogich komputerów pracujących z podziałem czasu, komputerów pochodzących z lat siedemdziesiątych ubiegłego wieku. Przy takim założeniu tajne oraz jawne programy pracowały na tym samym, jednym, drogim komputerze centralnym. Dzisiejsze komputery są znacznie tańsze. Co więcej, model taki nie uniemożliwiłby przenikania wirusów między obszarami niskiego i wysokiego bezpieczeństwa. Jego intencją było uniemożliwienie wypływu jawnymi i tajnymi kanałami poufnych danych. W ogóle nie wzięto pod uwagę zagadnień sieciowych.

Nowsze, powstałe w innych krajach standardy miały szerszy zakres. W Wielkiej Brytanii ogłoszono w 1989 roku „Poziomy zaufania". W Niemczech, Francji, Danii oraz Wielkiej Brytanii stworzono *Kryteria oceny zabezpieczeń informatyki* (ang. *Information Technology Security Evaluation Criteria*), dokument opublikowany przez Komisję Europejską. Dokument ten oraz wydane w 1993 roku *Kanadyjskie kryteria oceny wiarygodnych urządzeń komputerowych* (ang. *Canadian Trusted Computer Product Evaluation Criteria*) doprowadziły do powstania szkicu Kryteriów Federalnych, które następnie stały się podstawą *Powszechnych kryteriów* [ang. *Common Criteria*, CC, 1999] przyjętych przez ISO.

„Powszechne kryteria", wolne od aspektów politycznych, mają zostać przyjęte we wszystkich krajach sygnatariuszach. W dokumencie tym próbuje się oddzielić różne aspekty bezpieczeństwa. W ten sposób, poza gwarancjami będącymi oddzielną skalą oceny (ktoś może dysponować systemem wyposażonym, w jakieś funkcje, mającym wysoką gwarancją bezpieczeństwa lub systemem, z tymi samymi funkcjami, mającym niską gwarancję bezpieczeństwa), zostały od siebie oddzielone różne funkcje. Dzięki temu niektóre bezpieczne systemy mogą obsługiwać szyfrowanie i zarządzać wykorzystywaniem zasobów, nie troszcząc się jednocześnie o wiarygodne ścieżki. To jednak wskazuje, że trudniej będzie zrozumieć, co dokładnie oznacza dla systemu bycie „bezpiecznym", gdyż trzeba również wiedzieć, do jakich zadań system ten został stworzony.

Innymi słowy, dobre zwyczaje inżynierii oprogramowania są wymuszane i egzekwowane przez instytucje weryfikujące. Niemniej jednak, jak wiemy, nawet najlepiej zaprojektowany system zawiera błędy.

Robak Morrisa (ang. *Morris Worm*) oraz wiele z jego dzisiejszych następców udzieliło nam szczególnie przydatnej lekcji, wskazując na istotę sprawy. Skutek błędu nie koniecznie ogranicza się do szkodliwych efektów lub niepoprawnego działania konkretnej usługi, w której błąd się znajduje. Z powodu wadliwego składnika zaatakowany może zostać cały system. Oczywiście, nikt nie pisze kodu z założenia zawierającego błędy, dlatego nie ma na idealnego środka ochronnego, ale są kroki, które można podjąć, żeby zwiększyć szanse powodzenia.

Pierwszy rzecz to paranoidalne podejście do pisania oprogramowania serwerów sieciowych. Hakerzy i tak zaatakują, dlatego trzeba się na to odpowiednio przygotować. Nie należy wierzyć, że cokolwiek, co zostało wysłane, jest w jakiś sposób poprawne lub nawet przydatne. Poprawność wszystkich danych wejściowych należy sprawdzać po każdym względem. Jeżeli program korzysta z jakichkolwiek buforów o stałej długości (nie tylko buforów wprowadzania danych), należy się upewnić, że bufory się nie przepełnią. Stosując dynamiczne przydzielanie pamięci, co z pewnością stanowi dobry pomysł, należy przygotować się na wyczerpanie wolnej pamięci lub wyczerpanie zasobów systemu plików oraz pamiętać, że mechanizmy wyjścia z takich sytuacji też mogą potrzebować pamięci oraz przestrzeni na dysku.

Dodatkowo, należy precyzyjnie zdefiniować składnię wejściową. Nie można sprawdzać poprawności czegokolwiek, gdy nie wiadomo, co jest poprawne. Z wielu przyczyn dobrym pomysłem jest stosowanie narzędzi do pisania kompilatorów, takich jak *yacc* czy *lex*, a najważniejszą z nich jest ta, że nie można napisać gramatyki wejściowej nie *wiedząc*, co jest dopuszczalne. Wówczas jest się zmuszonym do stworzenia jasnych definicji wzorców danych wejściowych możliwych do zaakceptowania. Widzieliśmy zbyt wiele programów nie wytrzymujących naporu niedorzecznych danych, których autor programu się nie spodziewał. Znacznie lepszym wynikiem działania programu jest pojawienie się komunikatu „błąd składni".

Następną zasadą jest *jak najmniej uprawnień* (ang. *least privilege*). Nie należy dawać demonom więcej możliwości, niż to niezbędne. Bardzo niewiele z nich musi działać z uprawnieniami root, zwłaszcza w komputerach stanowiących zapory sieciowe (*firewall*). Na przykład, część lokalnego systemu dostarczania poczty potrzebuje szczególnych uprawnień w celu przekopiowania wiadomości wysłanej przez jednego użytkownika do skrzynki pocztowej innego użytkownika. Przekaźnik poczty bramy pocztowej nie wykonuje takiego działania. Zamiast tego kopiuje pocztę z portu jednej sieci do portu drugiej sieci, a to zupełnie różne działania.

Nawet w przypadku serwerów, gdy *wydaje się*, że potrzebują specjalnych uprawnień, nie musi to być prawdą, gdy są one poprawnie skonstruowane. Serwer FTP systemu Unix, żeby przytoczyć tu najbardziej rażący przykład, wykorzystuje uprawnienia użytkownika *root*, by umożliwić logowanie się użytkowników oraz by powiązać kanał danych z portem 20. Tego drugiego wymagania nie można ominąć w całości, gdyż wymaga tego protokół, ale dokonanie kilku zmian umożliwiłoby małemu, prostemu i co bardziej oczywiste — mającemu poprawne uprawnienia programowi — wykonanie tych i tylko tych czynności. Z problemem logowania mógłby podobnie poradzić sobie program pracujący między użytkownikiem a właściwym programem, który przetwarza tylko polecenia *USER* oraz *PASS*, tworzy poprawne środowisko, oddaje swoje uprawnienia i wywołuje *nieuprzywilejowany* program realizujący resztę protokołu. Opis takiego programu znajdzie się w podrozdziale 8.7.

Nie należy poświęcać poprawności programu, a w tym weryfikowania poprawności, w pogoni za wydajnością. Jeżeli po to, by zaoszczędzić kilka nanosekund, program musi być złożony, skomplikowany, musi działać w sposób uprzywilejowany lub łączyć te działania, to najprawdopodobniej został on źle zaprojektowany. Poza tym urządzenia stają się coraz tańsze i coraz szybsze, a czas, który trzeba przeznaczyć na pozbycie się intruzów, oraz czas użytkowników pozbawionych usług jest drogi i staje się coraz droższy.

5.4. Niepowodzenia uwierzytelniania

Доверяй но проверяй — „Ufaj, ale sprawdzaj"

Przysłowie rosyjskie

Przyczyny ataków, które do tej pory opisaliśmy, wiążą się z uchybieniami mechanizmu uwierzytelniania. Rozumiemy przez to, że mechanizm, który dotąd był wystarczający, w jakiś sposób został pokonany. Sprawdzanie poprawności adresu źródłowego może w określonych warunkach działać poprawnie (na przykład, kiedy zapora sieciowa odrzuca fałszerstwa), ale hakerzy mogą posłużyć się programem *rpcbind* i retransmitować niektóre żądania. Wówczas końcowy serwer zostaje zmylony, ponieważ mu wydaje się, że pojawiające się komunikaty są pochodzenia lokalnego, a w rzeczywistości pochodzą skądinąd.

Uwierzytelnianie oparte na adresach zawodzi również, gdy komputer źródłowy nie jest wiarygodny. Oczywistym tego przykładem są komputery PC. Mechanizm, który został wymyślony w czasach, kiedy komputery pracujące z podziałem czasu były normą, przestał się sprawdzać, gdyż każdy może zarządzać własnym komputerem. Oczywiście, opcja w postaci haseł też nie zapewnia już bezpieczeństwa w sieciach składających się z takich komputerów, w których podsłuchiwanie haseł stało się łatwe i powszechne.

Czasem uwierzytelnianie nie udaje się, gdyż protokół nie zawiera właściwych informacji. Ani TCP, ani IP nie identyfikują nigdy nadającego użytkownika (jeżeli w ogóle w komputerach funkcjonuje taka możliwość). Protokoły, takie jak X11 czy *rsh*, muszą informację tę zdobyć same lub poradzić sobie bez niej (a jeżeli mogą ją uzyskać, to muszą znaleźć sposób na bezpieczne przeniesienie jej przez sieć).

Nawet kryptograficzne uwierzytelnienie komputera źródłowego może nie być wystarczające. Jak wspominaliśmy wcześniej, komputer, do którego się włamano, nie może dokonywać wiarygodnego szyfrowania.

5.4.1. Wyścigi do uwierzytelnienia

Podsłuchiwacze mogą łatwo przechwycić zapisane czystym tekstem hasło wysłane w nieszyfrowanej sesji, ale mogą też natknąć się na jeden ze schematów haseł jednorazowych[2]. Poprawnie działający schemat uwierzytelniania, stosujący hasła jednorazowe dla następnego logowania, niezależnie od miejsca jego pochodzenia, przeznacza tylko jedno prawidłowe hasło. Dobrym tego przykładem jest następne hasło na liście OTP, która zostanie opisana w podrozdziale 7.5, będące pierwszym znanym celem opisanego tu ataku.

W przykładzie tym założymy, że hasło ma znaną długość i składa się z samych cyfr. Napastnik inicjuje dziesięć połączeń z odpowiednią usługą. Każde połączenie czeka na to samo nieznane hasło. Prawidłowy użytkownik łączy się i zaczyna wpisywać poprawne hasło. Program atakujący obserwuje to i przekazuje do dziesięciu swoich połączeń poprawne znaki hasła w miarę ich wpisywania przez użytkownika. Gdy do wpisania pozostaje tylko jedna cyfra, program wysyła dziesięć różnych cyfr do wszystkich swoich

[2] *http://www.tux.org/pub/security/secnet/papers/secureid.pdf — przyp. aut.*

połączeń zanim użytkownik zdąży wpisać ostatnią cyfrę. Komputer jest szybszy, zatem wygrywa wyścig i jedno z połączeń napastnika zostanie uznane za poprawne. Takie schematy uwierzytelniania umożliwiają jedynie jednorazowe zalogowanie się z wykorzystaniem każdego hasła, więc właściwy użytkownik zostanie odrzucony i będzie musiał spróbować jeszcze raz. Oczywiście w takiej sytuacji napastnik będzie musiał znać długość hasła, ale zwykle długości te są dobrze znane.

Jeżeli napastnik w czasie uwierzytelniania umieści się między klientem a serwerem, może wygrać uwierzytelnione połączenie z komputerem dzięki przekazaniu klientowi wezwania pochodzącego od komputera i poznaniu od klienta poprawnej odpowiedzi. Atak na jeden z takich protokołów został opisany w pracy Bellovina i Merritta [1994].

Uwierzytelniający się może zrobić wiele, by udaremnić tego rodzaju atak [Haller *et al.*, 1998], ale te działania będą tylko łatami na wrodzoną słabość stosowanego schematu uwierzytelniania. Uwierzytelnianie metodą wezwanie-odpowiedź zupełnie udaremnia tego rodzaju atak, gdyż każde połączenie napastnika uzyskuje inne wezwanie i wymaga innej odpowiedzi.

5.5. Błędy protokołów

W poprzednim podrozdziale omówiono przypadek, w którym wszystko działało, tak jak należy, a niemożliwe było tylko wiarygodne uwierzytelnienie. Teraz weźmiemy pod uwagę odwrotność tamtej sytuacji, czyli warunki, w których działające protokoły zawierają błędy lub są nieodpowiednie, a przez to uniemożliwiają właściwą pracę aplikacji.

Omawianym przypadkiem będzie atak na numer sekwencyjny protokołu TCP, opisany w rozdziale 2. Z powodu niedostatecznej losowości generowanego początkowego numeru sekwencyjnego połączenia, napastnik może posłużyć się fałszowaniem adresu źródłowego (ang. *spoofing*). Trzeba przyznać, że gdy wymyślano numery sekwencyjne protokołu TCP nie przewidywano konieczności odpierania złośliwych ataków. W zakresie, na którym bazuje uwierzytelnienie wykorzystujące adresy, definicja protokołu jest niewystarczająca. Inne protokoły polegające na numerach sekwencyjnych też mogą być podatne na ten rodzaj ataku. Lista jest bardzo długa, zawiera protokół DNS oraz wiele protokołów opartych na RPC.

W świecie kryptografii popularną zabawą jest wynajdywanie luk w protokołach. Czasem ich twórcy po prostu popełniali błędy. Częstą przyczyną luk są też przyjęte, różne założenia. Udowadnianie poprawności wymian kryptograficznych to trudna sztuka i przedmiot wielu badań naukowych. Póki co, luki pojawiają się zarówno w rozważaniach akademickich, jak i w świecie realnym, na co wskazują ponure wskazówki „Tych, którzy wiedzą" (ang. *Those Who Know*).

Bezpieczne protokoły muszą być zbudowane na bezpiecznych podstawach. Weźmy pod uwagę świetny (mamy nadzieję, że świetny) protokół bezpiecznego zdalnego dostępu *ssh*. Protokół *ssh* ma funkcję, dzięki której użytkownik może podać wiarygodny klucz publiczny, przechowując go w pliku *authorized_keys*. Następnie, gdy klient zna prywatny klucz, użytkownik może zalogować się bez potrzeby wpisywania hasła. W systemie Unix plik ten zwykle przechowywany jest w katalogu *.ssh*, w katalogu domowym użytkownika. Rozważmy teraz sytuację, w której ktoś za pomocą *ssh* loguje się do komputera

z zamontowanymi przez protokół NFS katalogami domowymi. W takim środowisku napastnik może podszyć się pod odpowiedzi NFS i wprowadzić fałszywy plik *authorized_keys*. Zatem bezpieczeństwo postrzeganego jako wiarygodny protokołu *ssh* zawodzi w niektórych dość powszednie stosowanych środowiskach.

Plik *authorized_keys* charakteryzuje się jeszcze jedną drobną wadą. Kiedy użytkownikowi w nowym środowisku założone zostanie nowe konto, kopiuje on tam zwykle ze swojego starego konta wszystkie ważne pliki. Nie jest niczym niezwykłym to, że użytkownik przekopiuje też cały katalog *.ssh*, wobec czego wszystkie klucze *ssh* staną się dostępne w nowym koncie. Jednak użytkownik może nie zdawać sobie sprawy, że kopiując plik *authorized_keys* powoduje, że do nowego konta możliwy będzie dostęp za pomocą kluczy uwiarygodnionych do korzystania z konta poprzedniego. I chociaż może się to okazać drobnostką, to jest jednak możliwe, że nowe konto jest bardziej wrażliwe, a automatyczne nadanie do niego dostępu przez *ssh*, może nie być pożądane.

Zwróć uwagę, że w takiej sytuacji dostęp do konta przyznawany jest przez użytkownika, a nie przez administratora systemu, co zasadniczo nie jest dobrym pomysłem.

Innym przypadek stanowi błąd w protokole standardu sieci bezprzewodowych 802.11. Problemy z działaniem protokołu WEP (podrozdział 2.5) pokazują, że trudno zbudować system bezpieczeństwa i że inżynierowie, wykorzystujący do budowy systemu kryptografię, powinni zasięgnąć opinii kryptografów, a nie próbować tworzyć samemu całego system od początku. Bezpieczeństwo to dyscyplina bardzo specjalistyczna, dlatego nie ma w niej miejsca dla amatorów.

5.6. Wypływanie informacji

Większość protokołów niesie pewne informacje. Często też taka jest intencja, zebranie pewnych informacji, osoby korzystającej z ich usług. Witamy w świecie szpiegostwa komputerowego. Informacja sama w sobie może być celem szpiegostwa przemysłowego albo może być potrzebna jako środek do włamania się do systemu. Jednym z oczywistych przykładów jest protokół *finger*. Oprócz tego, że jest przydatny osobom odgadującym hasła, informacje przez niego dostarczane mogą być wykorzystywane w socjotechnice. Na przykład: „Witam Robin, wysiadła mi tu w East Podunk bateria mojego podręcznego mechanizmu uwierzytelniającego. Żeby wysłać Ci tę wiadomość musiałem skorzystać z cudzego konta. Czy możesz wysłać mi jego dane kodujące?" „Oczywiście nie ma problemu, wiedziałem, że jesteś w podróży. Dziękuję Ci za przysłanie harmonogramu."

Przydatne mogą być nawet tak przyziemne informacje, jak numer telefonu czy numer pomieszczenia. W czasie skandalu Watergate Woodward i Bernstein wykorzystali książkę telefoniczną komitetu reelekcji prezydenta do odtworzenia jego struktury organizacyjnej [Woodward i Bernstein, 1974]. Wątpliwości odnośnie tego, które informacje mogą być ujawniane, można potwierdzić u pracowników biura ochrony firmy, a ich praca polega na mówieniu „nie".

Na stronach internetowych niektórych firm udostępniane są ich książki telefoniczne. Jest to oczywiście bardzo wygodne, ale w świecie korporacyjnym informacje te często traktowane są jako poufne. Łowcy głów uwielbiają takie okazje. Przydają im się do poszukiwania osób mających określone kwalifikacje. Również uniwersytety podobnie traktują

takie informacje. Warunki ochrony prywatności (a często i surowość prawa) decydują o tym, jakie informacje mogą być udostępniane. Przykładem tego może być „Ustawa o prawie do edukacji i prywatności" (FERPA) oraz „Dyrektywy prywatności" Unii Europejskiej.

Innym owocnym źródłem danych jest system DNS. Opisaliśmy już bogactwo informacji, które można dzięki niemu zdobyć, począwszy od szczegółów organizacyjnych, a skończywszy na liście potencjalnie celów ataku. Kontrolowanie wypływu tych danych jest trudne i często jedynym wyjściem jest ograniczenie informacji podawanych przez widoczne z zewnątrz serwery DNS tylko do komputerów pełniących funkcje bram.

Doświadczeni hakerzy oczywiście wiedzą o tym i nie przyjmują do wiadomości podawanych im informacji o działających komputerach. Zamiast tego, w poszukiwaniu interesujących usług i ukrytych komputerów, skanują porty w danej przestrzeni adresowej. Najlepszym sposobem obrony jest wówczas dobra zapora sieciowa, gdyż maleje możliwość jego opanowania, jeżeli nie uda im się wysłać pakietów do komputera.

5.7. Ataki lawinowe — wirusy i robaki

W atakach lawinowych wykorzystywane są samorozprzestrzeniające się programy, do powielenia których dochodzi bardzo szybko. Programy potrafiące rozsyłać się same nazywane są *robakami* (ang. *worm*), natomiast programy dołączające się do innych programów — *wirusami* (ang. *virus*). Matematyczne funkcje ich rozprzestrzeniania są podobne, a różnice między nimi nie są ważne. Epidemiologia takich programów jest podobna do infekcji biologicznych.

Sukces działania tych programów leży w wykorzystywaniu powszechnych błędów lub zachowań występujących w wielkich populacjach podatnych na to programów lub użytkowników. Mogą się rozprzestrzenić na cały świat w ciągu kilku godzin, a, potencjalnie, nawet w ciągu kilku minut [Staniford, *et al.*, 2002; Rubin, 2001]. Rozprzestrzeniając się w dużych społecznościach, mogą powodować olbrzymie szkody ekonomiczne. Robak Melissa zatykał w niektórych firmach, pochodzące z firmy Microsoft, systemy poczty elektronicznej przez pięć dni. Różne robaki obciążały znacznie działanie całego Internetu. Zagrożenie to nie jest nowe ani nie ogranicza się jedynie do Internetu. Wirus „IBM Christmas Card" zablokował wewnętrzną bisynchroniczną sieć firmy IBM w 1987 roku [więcej szczegółów na ten temat można znaleźć w *RISKS Digest*, tom 5., numer 81.].

Programy tego rodzaju raczej korzystają z nadarzających się okazji, niż kierują się na określone osoby czy organizacje. Ale ich ładunek może trafiać i trafia w popularne cele polityczne i komercyjne.

Istnieje kilka sposobów na zminimalizowanie możliwości zarażenia się wirusem. Najmniej popularną metodą jest trzymanie się z dala od popularnych monokultur. Jest mało prawdopodobne, by zostać zarażonym, jeżeli używa się jedynie napisanych przez siebie systemów operacyjnych oraz aplikacji. Olbrzymia większość wirusów powstała dla systemów Windows firmy Microsoft, co oznacza, że użytkownicy systemów Unix oraz Macintosh ucierpieli mniej. Jednak ta sytuacja się już zmienia i to na niekorzyść użytkowników Linuksa. Obserwujemy obecnie dużo robaków Linuksowych oraz robaków niezależnych od platformy, rozprzestrzeniających się przez kilka monokultur, zarówno za pośrednictwem bezpośredniego dostępu do sieci, jak i stron WWW oraz poczty elektronicznej.

Nie można zarazić się wirusem, nie komunikując się z zarażonym komputerem. Ryzyko infekcji zmniejsza staranna kontrola połączeń sieciowych oraz plików otrzymywanych z obcych źródeł. Jest też wiele wirusów rozpowszechnianych dzięki ludziom przekazującym dalej swoim znajomym wiadomości pocztowe (często zawierające miejskie ciekawostki) z prośbą rozesłania ich dalej do kolejnych osób. W większości wypadków powoduje to tylko irytację, ale wśród osób o niewielkiej wiedzy na temat komputerów mogą one wywołać panikę. Niektóre z takich wiadomości zawierają nieprawdziwą informację o tym, że komputer odbiorcy tej wiadomości jest zarażony i niejednokrotnie instrukcję usunięcia pliku systemowego o istotnym dla działania systemu znaczeniu. Wiele osób postępując według tej instrukcji zniszczyło swoje systemy.

Oprogramowanie antywirusowe jest popularne i dość skuteczne w walce ze znanymi wirusami. Oprogramowanie to musi być stale uaktualniane, gdyż wyścig zbrojeń między piszącymi wirusy a firmami piszącymi oprogramowanie antywirusowe trwa stale. Wirusy potrafią nad wyraz skutecznie ukrywać swoją obecność i działania. Skanery antywirusowe nie mogą już tylko polegać na wyszukiwaniu określonych łańcuchów bajtów w plikach wykonywalnych, ale muszą emulować ich kod i umieć wyszukiwać śladów działania wirusów. W miarę jak wirusy będą stawały się coraz bardziej skomplikowane, oprogramowanie antywirusowe będzie musiało prawdopodobnie poświęcać coraz więcej czasu na przebadanie każdego pliku, a to w końcu może trwać zbyt długo. Może się również okazać, że twórcy wirusów będą pisać kody wirusów tak, by nie mogły one zostać zidentyfikowane w rozsądnym czasie.

Najlepszym rozwiązaniem byłoby uruchamianie tylko pewnych, niezmodyfikowanych programów. Mogą do tego celu zostać użyte technologie kryptograficzne, ale tego typu podejście jest obecnie hamowane polityczną wrzawą na temat mechanizmów ochrony praw autorskich oraz prywatności.

5.8. Ataki blokady usług

> „Halo! Halo! Czy jest tam kto? Halo! Zadzwoniłem, żeby powiedzieć cześć! Mówię halo! Słyszysz mnie Joe?
>
> Nie słyszę Cię. Wcale Cię nie słyszę. To niedobrze i wiem, dlaczego tak jest. Mysz przegryzła kabel. Żegnaj!"
>
> *Jedna ryba, dwie ryby, ryba czerwona, ryba niebieska*
> *Dr Seuss*

Omówiliśmy całą gamę popularnych ataków na komputery podłączone do Internetu. Ataki te wykorzystywały: słabości protokołów, błędy oprogramowania serwerów, a nawet nadgorliwość usłużnych ludzi. Ataki *odmowy usługi* (ang. *Denial of Service*), w skrócie DoS, to zupełnie co innego. Polegają one na nadużywaniu usług, czyli obciążaniu oprogramowania, sprzętu czy łączy sieciowych poza granice ich założonej wydajności. Celem takiego postępowania jest w najlepszym razie uniemożliwienie działania jakiejś usługi lub obniżenie jakości jej działania.

Te ataki różnią się też tym, że są one oczywiste, a nie są subtelne. Uniemożliwienie działania usługi powinno być łatwe do wykrycia. Owszem, ten rodzaj ataku łatwo zauważyć, ale za to trudniej ustalić jego źródło. Do ataku wykorzystywane są często pakiety zawierające sfałszowane, losowe wybrane (wobec czego bezużyteczne) adresy źródłowe.

Do przeprowadzenia *rozproszonego ataku odmowy usługi* (ang. *Distributed Denial of Service*) w skrócie DDoS, wykorzystywanych jest wiele komputerów włączonych do Internetu. Najczęściej komputery te uczestniczą w ataku zupełnie bezwiednie, gdyż w jakiś sposób wcześniej włamano się i umieszczono w nich złośliwy program. Bardzo trudno jest uwolnić się od ataku DDoS, gdyż atak następuje ze wszystkich stron. Poświęcono im cały punkt 5.8.3.

Na atak blokady usług nie ma niezawodnego lekarstwa. Tak długo, jak usługa jest dostępna ogólnie, ogół może robić z niej niewłaściwy użytek. W ten sposób, zupełnie anonimowo, można przez znaczny czas uniemożliwić działanie doskonale zabezpieczonego ośrodka WWW.

Łatwo wyliczyć szacunkową wartość ataku DoS. Firmy potrafią określić, ile może je kosztować kilkudniowe wstrzymanie działania serwera WWW z powodu ataku. Jeżeli nie potrafią tego zrobić, może to oznaczać, że nie mają dobrego planu biznesowego, w którym priorytetem jest wykorzystanie usług WWW.

Jeżeli uda się wykryć sprawcę ataku, firmy mogą dochodzić na drodze sądowej zrekompensowania swoich strat. Fakt ataku jest oczywisty i łatwo zrozumiały dla sądu. W ten sposób można też zmusić do współpracy różne pośrednio zaangażowane w to strony, takie jak na przykład dostawcy usług internetowych — ISP. Problemem oczywiście pozostaje znalezienie osoby, którą należy pozwać, gdyż ataki DDoS trudno jest prześledzić.

5.8.1. Ataki na łącza sieciowe

Ataki na łącza sieciowe mogą przebiegać pod różnymi postaciami, począwszy od prostego zasypywania wiadomościami pocztowymi (*mail bombing* lub *spamowanie*[3]), po wysyłanie pakietów przygotowanych specjalnie, by załamać działanie programu w atakowanym komputerze. Skutkiem takiego ataku może być: przepełnienie dysku, przeciążenie procesora, załamanie się systemu operacyjnego lub po prostu przeciążenie łącza.

Najbardziej prymitywnym atakiem tego rodzaju jest atak przeciążenia łącza sieciowego. Żeby przeciążyć łącze, atakujący musi wygenerować więcej pakietów, niż może obsłużyć odbiorca. W tych pakietach prawidłowe musi być tylko pole adresu docelowego, pozostała część może być przypadkowa (oczywiście, musi się też zgadzać suma kontrolna). By wypełnić na przykład łącze T1, wcale nie potrzeba tak dużo pakietów, wystarczy ich mniej niż 200 kb/s. Atak taki można przeprowadzić z pojedynczego komputera, pod warunkiem, że jego połączenie z siecią jest odrobinę szybsze niż połączenie celu ataku.

[3] Nazwa „spam" nie powinna się kojarzyć ze znakomitymi wyrobami mięsnymi firmy Hormel Corporation — *przyp. aut.*

Kilku napastników może przeprowadzić wspólny atak, koncentrując na celu ataku kilka generatorów pakietów. Ruch każdego generatora z osobna może być niewielki, ale suma ruchu musi być większa niż pojemność łącza sieciowego celu przeprowadzanego ataku. Gdy atak jest poprawnie koordynowany, jak to ma miejsce w przypadku ataków DDoS, setki komputerów korzystających z powolnych łączy może nawet przeciążyć łącze dużej przepustowości atakowanego komputera. Pozostawienie swojego adresu e-mail na popularnej stronie WWW, takiej jak Slashdot, może spowodować atak zasypywania wiadomościami pocztowymi po tym, jak spamerzy wejdą w posiadanie tego adresu.

5.8.2. Atak na warstwę sieciową

Wiele z najgorszych ataków przeprowadzanych jest na warstwę sieciową, czyli na działającą w komputerze implementację protokołu TCP/IP. Tego rodzaju ataki wykorzystują słabości lub błędy tego protokołu. Biorąc pod uwagę to, że typowa implementacja protokołu TCP/IP składa się z dziesiątek tysięcy wierszy kodu napisanego w języku C, wykonywanego w większości komputerów w obszarze uprzywilejowanym, trudno jest jego twórcy przygotować kod na wszystkie potencjalne niebezpieczeństwa. Cykl edycji, kompilacji i uruchamiania jest długi, a w przypadku protokołów niezwykle trudno jest śledzić ich działanie, a już szczególnie sytuacje, w których powstają błędy.

Może to w czasie normalnej pracy być przyczyną kłopotów. Jednak dużo większe problemy mogą wystąpić, gdy przeciwnik aktywnie poszukuje luk w działaniu protokołu lub tworzy pakiety, które są w stanie załamać cały system.

Pakiety zabójcy i pakiety ICMP

Przez lata w Internecie krążyło wiele plotek o dysponujących większymi możliwościami pakietach, czyli o bardziej złośliwych. Widzieliśmy już *pakiety zabójców* (ang. *killer packet*), które po trafieniu na błąd powodowały załamanie się systemu. Pakiety te mogą mieć bardzo duże rozmiary, być źle pofragmentowane, zawierać bezsensowne opcje lub inne atrybuty, które powodują zadziałanie części niezbyt często używanego kodu [przykłady takie można znaleźć w poradniku CERT Advisory CA-96-26 z 18 grudnia 1996 oraz CERT Advisory CA-00-11 z 9 stycznia 2000]. Napastnicy specjalizujący się w algorytmach dzięki wykorzystaniu słabości mechanizmu kolejkowania lub wyszukiwania (jeden z takich przypadków omówiony zostanie dalej w tym rozdziale) potrafią spowodować, że programy zaczną działać niewydolnie.

Niektórzy ludzie uwielbiają podsyłać ośrodkom sieciowym fałszywe pakiety ICMP zakłócające ich komunikację. Czasem są to komunikaty *Destination Unreachable*, a czasem wprowadzające więcej zamieszania i bardziej śmiercionośne komunikaty na przykład zerujące maskę sieci komputera. Czemu komputery odbierają komunikaty, o które nie wysyłały zapytań? Inni hakerzy zabawiają się protokołami routingu, nie w celu opanowania maszyny, tylko uniemożliwienia jej komunikacji z innymi.

Ataki pakietami SYN

Niektóre pakiety uderzają w cel z większą siłą niż inne. Pierwszy dobrze opisany atak odmowy usług został przeprowadzony na dostawcę usług internetowych — firmę Panix. Do komputera firmy Panix kierowano w ciągu sekundy około 150 pakietów TCP SYN

(więcej szczegółów można znaleźć w punkcie 2.1.3). Pakiety te zasypały mechanizm przetwarzania połączeń półotwartych jądra systemu Unix, co okazało się całkiem prostą czynnością. Gdy tabela połączeń półotwartych wypełniła się, nadchodzące od upoważnionych użytkowników polecenia nawiązania połączenia były odrzucane, uniemożliwiając im korzystanie z komputera. Ataki z wykorzystaniem pakietów SYN opisane są bardziej szczegółowo w pracy Northcutta i Nova [2000].

Jest to jedyny atak, o którym nie wspomnieliśmy w pierwszym wydaniu tej książki, gdyż nie znaliśmy żadnych metod jego odparcia. Opis tego ataku został usunięty w ostatniej chwili, gdy książka była już kierowana do druku. Teraz żałujemy tamtej decyzji. Atak na firmę Panix został przeprowadzony z użyciem programu, które dwa pisma hakerskie opublikowały kilka miesięcy wcześniej [deamon9 *et al.*, 1996].

To, że dość mały strumień pakietów mógł uniemożliwić działanie określonej usługi TCP komputera, spowodowane było tym, że oprogramowanie TCP większości systemów nigdy nie było pisane z myślą o odpieraniu tego rodzaju ataku. To dobry przykład siły pakietów wobec słabości programu. W wyniku tego ataku znacząco wzmocniono właściwy program TCP.

Spam — ataki na poziomie aplikacji

Możliwe jest oczywiście przeciążenie komputera na poziomie jego aplikacji. Celem tego ataku może być wyczerpanie się tabeli procesów lub dostępność procesora.

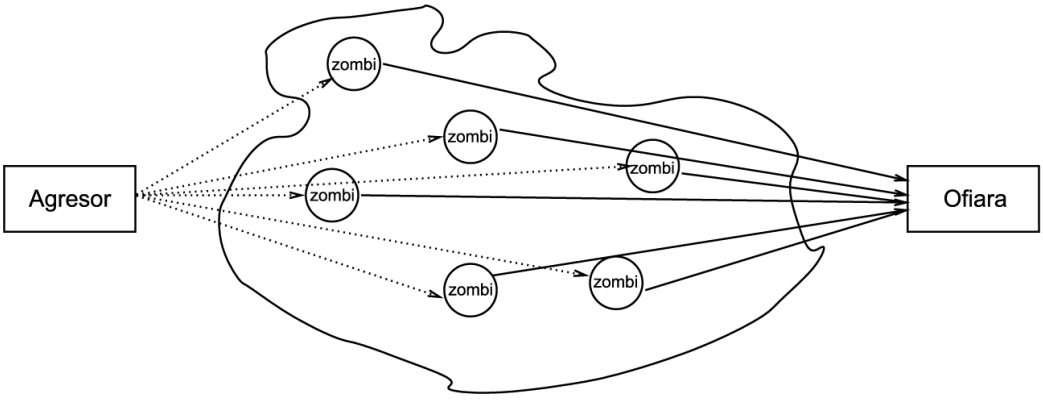
Wysyłając kilka gigabajtów danych za pomocą poczty elektronicznej czy FTP, można prawdopodobnie wypełnić nimi cały dysk. Trudno bowiem ustalić nieprzekraczalny, górny próg użycia zasobów. Nie trzeba być wcale uprawnionym, zaawansowanym użytkownikiem, by w prosty sposób wysłać kilkaset razy po 1 MB danych. Dodatkowo, działanie to spowoduje utworzenie w komputerze bardzo wielu procesów odbierania danych, co obciąży go jeszcze bardziej.

Najlepsze, co w takiej sytuacji można zrobić, to zapewnić wystarczającą ilość zasobów do obsłużenia wszystkich potrzeb (obecnie ceny dysków gwałtownie spadają) we właściwych miejscach (na przykład oddzielnych obszarach dla poczty, usług FTP i, szczególnie cennych, danych raportów zdarzeń) oraz zabezpieczyć się przed łagodnymi błędami. System pocztowy, który nie może przyjąć i umieścić w kolejce całej przesyłki, powinien poinformować o tym jej nadawcę. Nie powinien on komunikować o powodzeniu operacji zanim nie ma pewności, że odesłanie przesyłki zakończyło się powodzeniem.

Spam pocztowy jest faktem. Większość użytkowników Internetu otrzymuje każdego dnia porcję tego typu wiadomości i to po tym, jak ich dostawcy usług internetowych odfiltrowali już co bardziej oczywiste „śmieci". Ogrom tego problemu stał się dla nas jasny, gdy założyliśmy konto pocztowe na jednym z darmowych serwerów i posłużyliśmy się nim przy sprzedaży pewnej rzeczy na aukcji internetowej. Chociaż nie używaliśmy tego konta do innych celów, to za każdym razem, gdy do niego zaglądaliśmy (około raz na miesiąc), znajdowaliśmy w nim setki niezamówionych wiadomości pocztowych, informujących o przeróżnych stronach WWW dotyczących odchudzania, szybkiego robienia pieniędzy i spełniania innych fantazji. Dla większości ludzi spam jest niedogodnością, którą zdążyli zaakceptować. Jednak spam powodowany przez wirusy poczty elektronicznej oraz robaki (a także ludzi, którzy powinni lepiej zdawać sobie sprawę z tego, co robią) zniszczył już wiele systemów pocztowych.

5.8.3. DDoS

Uwaga całego świata skierowana została na ataki typu DDoS po tym, jak w lutym 2000 roku uniemożliwiły one działanie jednego z bardziej znanych portali WWW. Przypadkowo stało się to w niedługim czasie od opisania przez jednego z nas, Steve'a, w *The North American Network Operators' Group* (NANOG) działanie takiego ataku. W dzienniku *Washington Post* zaczęto nawet doszukiwać się związku między tymi wydarzeniami. Wątpimy w taki związek, jednak nie wiemy na pewno, że nie miał miejsca. Ataki DDoS to odmiana ataków DOS przeprowadzana jednocześnie z wielu zwerbowanych do tego celu w całej sieci komputerów. Sposób ich działania przedstawiono na rysunku 5.1.



Rysunek 5.1. *Rozproszony atak odmowy usług. Napastnik wysyła komunikat do programu głównego. Program główny rozsyła komunikaty do zombich, którzy z kolei zasypują cel swoimi pakietami*

Kolejne kroki tego ataku przedstawiają się następująco:

1. Napastnik za pomocą odpowiednich środków instaluje program *zombi* na tak wielu komputerach, jak to tylko możliwe, w wielu różnych domenach administracyjnych. Program zombi wiąże się z jednym z portów i oczekuje instrukcji.
2. Napastnik instaluje program *główny* w jednym z komputerów w Internecie. Program główny dysponuje listą komputerów, na których zainstalowane są wszystkie programy zombi.
3. Napastnik czeka.
4. Kiedy nadchodzi czas ataku, napastnik wysyła programowi głównemu komunikat, w którym przekazuje mu adres celu ataku.
5. Wszystkie programy zombi jednocześnie wysyłają w kierunku celu taką liczbę pakietów, która go obezwładnia.

Wiadomość pochodząca z programu głównego do programów zombi ma zwykle sfałszowany adres nadawcy. W celu utrudnienia identyfikacji nadawcy może zostać użyta technika kryptograficzna. W celu utrudnienia wykrycia faktycznego źródła, pakiety wysyłane przez programy zombi też mogą mieć sfałszowane źródłowe adresy IP, chociaż większość napastników wydaje się tym nie przejmować. Dodatkowo, program główny umieszcza swoje komunikaty w pakietach ICMP odpowiedź echa, które są przepuszczane przez wiele zapór sieciowych.

W Internecie można znaleźć sporo popularnych narzędzi ataków DDoS oraz ich odmian. Jednym z pierwszych był program TFN (ang. *Tribe Flood Network*). Jest on dostępny w postaci źródłowej w wielu miejscach w Internecie. Napastnik może wybrać jedną z technik zasypywania pakietami, takich jak: zasypywanie pakietami UDP, pakietami TCP SYN, pakietami ICMP Echo Request lub może wybrać atak typu smurf. Program główny informuje programy zombi o rodzaju stosowanego ataku w komunikacie ICMP Echo Reply. Innymi narzędziami do przeprowadzania ataków DDoS są: TFN2K (bardziej zaawansowana wersja programu TFN, przeznaczona dla systemu Windows NT i różnych odmian Uniksa), Trinoo oraz Stacheldraht. Ten ostatni jest bardzo zaawansowanym programem, który zawiera funkcje szyfrowania swoich połączeń i funkcje automatycznego uaktualniania. Stąd już tylko krok do stosowania przez hakerów systemów PKI. Czy już nikomu w sieci nie można zaufać?

Nowsze narzędzia są nawet bardziej skomplikowane. Slapper, jeden z robaków systemu Linux, tworzy między podporządkowanymi sobie komputerami sieć równorzędną, co łagodzi pewne problemy w komunikacji z programem głównym. Inne programy wykorzystują do celów sterownia kanały IRC.

5.8.4. Co zrobić w przypadku rozproszonego ataku odmowy usług?

Z rozproszonymi atakami odmowy usług trudno sobie poradzić. Ataki takie można złagodzić, ale nie ma na nich całkowicie skutecznego środka.

> *Każda ogólnie dostępna usługa może zostać przez ten ogół nadużyta.*

Jeżeli komputer stanie się celem ataku tego rodzaju, można zrobić cztery rzeczy:

1. Znaleźć sposób na odfiltrowanie szkodliwych pakietów.
2. W jakiś sposób poprawić przetwarzanie nadchodzących danych.
3. Odnaleźć i uniemożliwić działanie ośrodka, z którego prowadzany jest atak.
4. Rozbudować sprzęt i zwiększyć pojemność sieci tak, by normalny ruch oraz ruch wywołany atakiem mógł zostać obsłużony.

Żadna z tych reakcji nie jest doskonała. Nieświadomie można rozpocząć wyścig zbrojeń z napastnikiem, a sukces w tym wyścigu zależy tylko od tego, jak daleko będzie chciał on się posunąć. Przyjrzyjmy się bliżej tym rozwiązaniom.

Odfiltrowywanie szkodliwych pakietów

W atakujących pakietach może być zawarty na tyle charakterystyczny element, który znacznie ułatwi ich odfiltrowanie. Być może pakiety te wysyłane są z określonego portu. Możliwe, że pochodzą z sieci, w której nie ma żadnego uprawnionego użytkownika. Te nietypowe szczegóły mogą mieć całkiem techniczny charakter. W jednym z ataków numery sekwencyjne pakietów TCP zawsze rozpoczynały się od tej samej wartości. Próbując odrzucać szkodliwe pakiety, może konieczne okazać się zagłębienie w szczegóły TCP oraz IP.

Filtr odrzucający pakiety można uruchomić na routerze lub nawet w jądrze atakowanego komputera. Filtr nie musi być doskonały, dopuszczalne jest odrzucenie przez niego jakiegoś procentu ruchu uprawnionego. Szczegóły przeciwdziałania zależą od rodzaju

ataku i sposobu działania firmy. Lepiej, jeśli pracować może 80% użytkowników niż 0% użytkowników. Nie jest to rozwiązanie idealne, ale w przypadku tego rodzaju ataku nie obiecywaliśmy rozwiązań doskonałych.

We wczesnej fazie ataku na firmę Panix numery sekwencyjne TCP nie były losowe, dzięki czemu w prosty sposób odfiltrowywano szkodliwe pakiety. Później napastnicy zmienili je na liczby losowe i wyścig zbrojeń trwał nadal. Adres źródłowy i losowy numer sekwencyjny atakujących pakietów były generowane za pomocą funkcji *rand* oraz *random*. Czy można przewidzieć pseudolosową sekwencję i na tej podstawie identyfikować atakujące pakiety? Gene Spafford odkrył, że jest to możliwe, pod warunkiem, że atakujący komputer korzysta ze słabego generatora liczb losowych. Jeden z publicznie dostępnych programów w trakcie przeprowadzania ataku wysyłał pakiety zawierające dość małą wartość początkową w polu TTL. Na tej podstawie mogliśmy odrzucać te pakiety, gdyż praktycznie wszystkie implementacje protokołu IP używają dość dużych wartości początkowych w polu TTL. Na tym etapie walki należy rozgrywać tego rodzaju gry, ale mając na uwadze to, że w tym samym czasie napastnicy udoskonalają swoje generatory pakietów. Trzeba pamiętać, że małe wartości TTL stosuje też w swoich pakietach program *traceroute*. Czy je też blokować?

Elastyczność Internetu — eksperci przychodzą na ratunek

Internet został stworzony tak, by mógł przetrwać uderzenie — pakiety potrafią omijać miejsce awarii. Dowiedzieliśmy się, że jedyną iracką siecią, która przetrwała ciężkie bombardowania w 1991 roku, była sieć pakietowa.

Farmerzy wiedzą, że uprawianie na dużym obszarze, takim jak stan Kansas, jednakowej odmiany pszenicy jest niebezpieczne. Nazywa się to *monokulturą*, a monokultury są podatne na pospolite ataki.

Internet jest prawie monokulturą. Żeby w nim działać, komputer musi używać jakiejś implementacji protokołu TCP/IP. Większość komputerów w Internecie korzysta z tej samej wersji tego samego oprogramowania. Jeżeli zostanie w nim wykryty błąd, będzie się on znajdować prawdopodobnie w milionach komputerów. Na tym właśnie polega podstawowa przewaga hakerów. Nieopłacalne i niemądre byłoby, gdyby każdy z nas pisał swój własny system operacyjny czy własną implementację protokołu TCP/IP.

Oznacza to również, że wielu ekspertów zna ten sam Internet i gdy pojawia się w nim nowe zagrożenie, mogą oni szybko zadziałać, wypracowując wspólną ekspertyzę dotyczącą nowych, interesujących problemów. Na myśl przychodzą nam dwa przykłady, choć z pewnością było też i wiele innych.

W 1988 roku pojawił się robak Morrisa i szybko zaatakował wiele ośrodków sieciowych. Kilka grup ekspertów natychmiast zdeasemblowało kod robaka, przeanalizowało go i opublikowało wyniki swoich badań. Szybko pojawiły się środki zaradcze i szczepionki, a rozprzestrzenianie się robaka zostało zahamowane w ciągu tygodnia.

Firma Panix została zaatakowana za pomocą ataku odmowy usług, wykorzystującego pakiety SYN. Grupa projektantów protokołu TCP/IP szybko stworzyła zamkniętą listę dyskusyjną, w której rozważano wiele możliwości rozwiązania tego problemu. Niedługo potem pojawił się przykładowy kod, który następnie został poddany dyskusji i w jej wyniku ulepszony. W ciągu tygodnia lub dwóch wielu producentów oprogramowania przygotowało łatki na swoje programy.

Społeczność internetowa korzysta z tego rodzaju współpracy. Nie zawsze uda się przewidzieć nowe zagrożenia, ale zawsze wielu ludzi gotowych jest opowiedzieć na nie i przygotować odpowiednie rozwiązania. Zwykle łatwo jest zainstalować nowy program, znacznie łatwiej niż ponownie zasiać cały stan Kansas.

Oczywiście, jeżeli problem leży w sprzęcie…

Występować też mogą jeszcze inne anomalie. Normalne pakiety mają pewne cechy charakterystyczne, których nie mają pakiety generowane losowo. Niektóre komercyjne programy poszukują tych anomalii i wykorzystują je do odrzucania atakujących pakietów.

Pakiety w typowym ataku mają losowy źródłowy adres IP. Gdyby zawsze był to ten sam adres lub gdyby były to adresy pochodzące z jakiegoś zakresu, można by je po prostu ignorować, pod warunkiem, że nie pochodziłyby one od jakiegoś ważnego klienta. Nawet gdy źródłowy adres IP jest adresem losowym, można spróbować odfiltrować kilka z nich, kierując się zupełnie sensownymi podstawami.

Przykładowo, mimo że większość z internetowej przestrzeni adresowej została rozdzielona, nie wszystkie adresy z tej przestrzeni są używane i nie wszystkie są dostępne z ogólnego Internetu. Firmie przyznano całą sieć /8, ale na zewnątrz może ona ogłaszać tylko małą jej część. Dzięki wykorzystaniu tej wiedzy, można odrzucać losowe pakiety, które sugerują, że pochodzą z pozostałej części takiej sieci.

Stworzenie mapy bitowej lub filtra Blooma [Bloom, 1970] dla 2^{24} nieprzydzielonych lub nieogłaszanych adresów nie byłoby rzeczą trudną. Od razu należy pominąć wszystkie sieci z zakresu multicastowego i wszystkie sieci, które nie znajdują się na liście sieci ogłaszanych przez protokół BGP4. Można by nawet za pomocą polecenia *ping* badać losowo niektóre z adresów i odmawiać obsługi pakietów przychodzących z nieodpowiadających sieci. W tym wypadku należy jednak zachować ostrożność, gdyż złe ustawienie w takiej tabeli jednego bitu mogłoby się stać świetnym atakiem blokady usług przeprowadzonym na samego siebie.

Oczywiście, mapy bitowej można by używać w całej sieci, a jej tworzenie mogłoby być dobrym rodzajem usługi. Nie twierdzimy, że filtr musi być koniecznie oparty na mapie, ponieważ istnieją jeszcze inne sposoby przeprowadzania takiej weryfikacji, w dodatku zajmujące mniejszą ilość pamięci. Tabele globalnego routingu wypełniają się do swoich granic, co powoduje konieczność rozbudowywania routerów.

Można też stworzyć filtr rozpoznający stałych użytkowników. W przypadku ataku można przejrzeć pliki raportów z ostatnich kilku miesięcy i wybrać z nich adresy stałych użytkowników oraz porty przez nich używane. Filtr sprawdza, czy pakiet pochodzi od przyjaciela, a jeśli nie, jest odrzucany.

Skuteczność tego rodzaju filtra zależy od rodzaju realizowanych usług. Lepiej będzie on funkcjonować, gdy stali klienci korzystają z sesji protokołu *telnet*, niż gdy szeroki ogół użytkowników korzysta z protokołu WWW. Na tej samej zasadzie można filtrować także ruch pocztowy. Wówczas wciąż odbierano by wiadomości pocztowe od stałych korespondentów, ale niestety tracone byłyby wiadomości pochodzące od nowych. Podkreślamy jeszcze raz, że filtrowanie nie jest doskonałym rozwiązaniem, ale dzięki niemu można prowadzić działalność przynajmniej częściowo.

W wolnym społeczeństwie jednostki zachowujące się niepoprawnie można zdyscyplinować za pomocą *unikania*. Możemy postanowić, że z niektórymi osobami nie rozmawiamy, i kropka. Różne grupy religijne, takie jak grupa Amish, wykorzystały to do wprowadzania w życie swoich zasad. Dzięki filtrom, które tu omówiliśmy, możemy odmówić korzystania z naszych usług tym, których nie lubimy.

Na przykład, jeżeli pakiety ataku odmowy usług stale nadchodzą z konkretnego uniwersytetu, można po prostu odciąć ten uniwersytet od korzystania z naszych usług. Taka sytuacja miała miejsce parę lat temu w przypadku uniwersytetu MIT, gdyż tak wielu hakerów wykorzystywało komputery tej uczelni, że wiele ośrodków sieciowych zaczęło odrzucać wszystkie pakiety przychodzące z tej uczelni.

Legalnym użytkownikom z uniwersytetu MIT utrudniono przez to korzystanie z usług wielu ośrodków. Spowodowało to, że na winnym wszystkiemu wydziale zmieniono zasady korzystania z Internetu, w wyniku czego większość hakerów przeniosła się gdzie indziej.

Czasem stosowna obrona jest uznawana za legalną. Było już kilka przypadków [na przykład CompuServe v. Cyber Promotions, Inc., 962 F.Supp. 1015 — S.D. Ohio 1997], w których sąd zabronił spamerowi irytowania klientów ISP. Pochwalamy takie decyzje.

Usprawnienie oprogramowania

Dysponując kodem źródłowym systemu, można go usprawnić. W większości ośrodków sieciowych rozwiązanie takie jest niepraktyczne, gdyż po prostu brak im czasu, wiedzy i chęci, aby modyfikować jądro do odpierania ataków odmowy usług. Kody źródłowe używanego oprogramowania są przeważnie niedostępne, jak w przypadku routerów lub produktów firmy Microsoft, wówczas ośrodki proszą o pomoc producentów lub szukają innych rozwiązań.

Ściganie

Jednak pakiety te skądś przecież przychodzą. Może da się wytropić ich źródło i zdusić atak. Nie dajemy zbyt wiele nadziei na schwycenie napastnika, gdyż prawie zawsze dokonuje on ataku za pomocą opanowanych przez siebie komputerów, ale może będziemy mieli trochę szczęścia.

Znajdujące się w pakietach pole TTL może udzielić wskazówek odnośnie liczby przeskoków między napastnikiem a tobą. Typowa ścieżka, po której poruszają się pakiety IP, może składać się z 20 lub więcej przeskoków, masz zatem do przebycia sporą odległość. Różne systemy operacyjne nadają temu polu charakterystyczne dla siebie wartości początkowe i w ten sposób może Ci się udać znacząco zawęzić zakres poszukiwań.

Adres źródłowy najprawdopodobniej w niczym nie pomoże. Jeżeli jest przewidywalny, to najprawdopodobniej najprostszym wyjściem będzie odfiltrowywanie tych pakietów i zignorowanie ich. Gdy adres źródłowy jest prawdziwy, skontaktowanie się ze źródłem powinno być łatwe, a w związku z tym możesz podjąć dalsze działania lub poskarżyć się zamieszanemu w to ISP. W przypadku ataku DDoS może to być niewykonalne z powodu zbyt dużej liczby różnych źródeł ataku.

Gdy adresy źródłowe są fałszowane i losowe, musisz prześledzić drogę pakietów przez ruchliwy szkielet Internetu, aż do jego źródła [Savage *et al.*, 2000]. Do tego potrzebne jest zrozumienie i współpraca ze strony ISP. Wiele ISP ciągle udoskonala swoje możliwości spełnienia tego warunku.

Czy ISP można namówić na współpracę? Większość można, gdy dostarczy się im postanowienie sądu, ale to jest utrudnione przez granice państwowe.

Czy przeprowadzenie takiego śledzenia jest zgodne z prawem? Czy to nie jest już podsłuchiwanie? Czy masz prawo zobaczyć kierowany do siebie pakiet, zanim on osiągnie twoją sieć?

Oczywiste rozwiązanie to wykorzystanie przez ISP poleceń routerów informujących o pojawieniu się określonych pakietów. Routery firmy Cisco zawierają polecenie `IP DEBUG`, za pomocą którego można wychwycić pakiety pasujące do odpowiedniego wzoru. Funkcję tę można uruchamiać na kolejnych routerach ISP, aż pakiety doprowadzą do ich klienta lub następnego ISP. Dowiedzieliśmy się, że użycie tego polecenia może spowodować zawieszenie się bardzo obciążonego routera. To samo należy zrobić w miejscu poprzedniego przeskoku, najprawdopodobniej u innego ISP i możliwe, że w innym kraju.

Niektóre routery wyposażone są w inne pomocne funkcje. Na przykład funkcja `NetFlow` w routerach firmy Cisco informuje o interfejsie, z którego nadchodzi dany ruch.

Stone [2000] opisuje sieć nakładkową, dzięki której można uprościć śledzenie źródła pakietów w ISP, ale wymaga to od ISP wyprzedzającego planowania.

Gdy atakujące pakiety nadchodzą od jednego z klientów ISP, ISP może się z nim skontaktować i prosić o pomoc lub też zainstalować filtr uniemożliwiający temu klientowi fałszowanie pakietów. Taki filtr jest rzeczywiście bardzo dobrym rozwiązaniem i niektórzy ISP instalują go na swoich routerach. Dzięki temu filtrowi pakiety przychodzące od klienta mają adres źródłowy pasujący do ogłaszanej dla niego sieci.

Filtr taki może spowolnić nieznacznie działanie routera, ale połączenia z klientami zwykle realizowane są za pomocą dość powolnych łączy. Typowy router może przy tych prędkościach filtrować, pozostawiając jeszcze procesorowi zapas mocy. Większym kłopotem jest dodatkowa praca administracyjna. Gdy ISP ogłasza nową sieć, musi też dokonać zmian w filtrze routera brzegowego. To dodatkowa praca i jeszcze jedna okazja do pomyłki.

Filtr taki nie powinien jedynie *odrzucać* sfałszowanych pakietów, gdyż stanowią one zbyt cenną informację, by się ich po prostu pozbyć. Fakt odrzucenia pakietu również powinien być gdzieś *odnotowywany*, a klient powinien zostać poinformowany, że generuje podejrzane pakiety. Takie działanie może pomóc w ujęciu hakera i uniemożliwieniu mu nadużywania komputera klienta. Może też być świadectwem kompetencji, której brak konkurencyjnemu ISP.

Byłoby miło, gdyby routery pracujące w rdzeniu Internetu również dokonywały podobnego filtrowania, odrzucając pakiety, które zawierają niewłaściwe adresy źródłowe. Mają przecież odpowiednie informacje (z tabel routingu BGP4), a sprawdzanie mogłoby być wykonywane równolegle z przeprowadzaniem routingu. Problemem jest tu jednak asymetryczność wielu ścieżek routingu. Takie rozwiązanie spowodowałoby zwiększenie złożoności i kosztów routerów, które i tak są już duże i drogie. Ani producenci routerów, ani ISP nie są zachęcani do dodania takiego filtrowania.

Są jeszcze inne sposoby wykrycia źródła, z którego nadchodzą sfałszowane pakiety. ISP może odłączać swoje główne łącza na kilka sekund i obserwować, czy pakiety przestają trafiać do celu. Tę prostą i brutalną metodę możesz stosować mając fizyczny dostęp do kabli. Większość klientów nie zauważy krótkiej przerwy. Kable należy odłączać tak długo, aż trafi się na właściwe łącze.

To samo działanie możesz podjąć, nie dysponując dostępem do kabli, za pomocą różnego rodzaju poleceń routera. Sugerowano również, by bardziej skorzy do współpracujący ISP ogłaszali trasę do zaatakowanej sieci, utrzymując w ten sposób pakiety z daleka od sieci mniej chętnych do współpracy. Jeżeli ten mechanizm nie zostałby zaimplementowany poprawnie, również mógłby stać się źródłem ataków odmowy usług.

Można wyobrazić sobie wydane routerowi polecenia „nie kieruj do mojej sieci pakietów przez jedną sekundę". W ten sposób można by spowodować przerwę w strumieniu nadchodzących pakietów i dzięki temu wyśledzić ich źródło. Tego polecenia mógłbyś użyć do zapoczątkowania ataku odmowy usług. Możliwe, że należałoby zabezpieczyć to polecenie podpisem kryptograficznym lub spowodować, by router przyjmował takie polecenia nie częściej niż co kilka minut. W tego typu gry możesz bawić się przy konfiguracji routera i z protokołami routingu, ale zapoczątkować je może tylko personel techniczny ISP[4].

Obiecującym podejściem do sterowania przeciążeniami jest *Pushback* [Mahajan *et al.*, 2002; Ioannidis i Bellovin, 2002]. Pomysł ten polega na identyfikowaniu przez router wzorców ruchu odpowiedzialnego za przeciążenie. Po zidentyfikowaniu, ruch taki jest odrzucany. Ostatecznie, żądania ponowienia prawidłowego ruchu są propagowane wstecz do jego źródła. Celem tego jest poprawa usług dla poprawnie zachowujących się strumieni współdzielących łącza razem z ruchem złym.

Rozbudowa atakowanego komputera

Jest to prawdopodobnie najskuteczniejsze lekarstwo na ataki blokady usług. Może też być i najdroższe. Gdy sieć jest zasypywana pakietami, możesz po prostu zainstalować szersze wejście. Szybszy procesor oraz zwiększona pamięć obsłużą większy ruch. W czasie ataku na firmę Panix wysunięto propozycję wprowadzenia zmiany do protokołu TCP tak, by stan połączeń półotwartych trwał krócej lub by TCP działał inaczej w ramach obowiązujących zasad.

Zwykle trudno szybko zwiększyć pojemność łącza sieciowego, a na dodatek jest to droga inwestycja. Zniechęca do tego też fakt, że duża ilość pieniędzy wydawana jest tylko z powodu ataku.

Najłatwiejsze może okazać się zwiększenie możliwości serwera. Komercyjne systemy operacyjne i oprogramowanie serwerów sieciowych znacząco różnią się między sobą pod względem wydajności. Pomocny w tym może być rozsądny wybór oprogramowania. Nie polecamy tu konkretnych producentów, chcemy tylko zauważyć, że implementacje z dłuższym stażem na rynku zwykle są silniejsze i wydajniejsze. Reprezentują sobą większą skumulowaną wiedzę.

[4] Więcej informacji można znaleźć w dokumencie *http://www.nanog.org/mtg-0210/ispsecure.html*, zwłaszcza na stronach 68 – 76 — *przyp. aut.*

W ten sposób jednak problem nie zniknie. Pewnego dnia w przyszłości, gdy wszystkie łącza sieciowe będą szyfrowane, wszystkie klucze rozdzielone, wszystkie serwery uwolnione od błędów, wszystkie komputery bezpieczne, a wszyscy użytkownicy uwierzytelnieni — ataki odmowy usług wciąż będą możliwe. Dobrze przygotowani decydenci zaaranżują dobrze nagłośnione ataki na popularne cele, takie jak rządy, największe firmy czy niepopularne osoby. Sądzimy, że takie ataki staną się kiedyś w Internecie codziennością.

5.8.5. Rozpraszanie wsteczne

Pakiet IP musi mieć adres źródłowy, to pole nie jest opcjonalne. Napastnicy stosujący atak DoS nie używają własnych adresów ani adresów szablonowych, gdyż mogą one pomóc w ujawnieniu źródła ataku lub co najmniej ułatwić identyfikację i odfiltrowanie takich pakietów. Często zamiast tego używają adresów źródłowych wybieranych losowo, dzięki czemu można łatwiej zmierzyć tempo ataków w skali całego Internetu.

Komputer atakowany pakietami DoS radzi sobie z obsługą części z nich. Wysyła odpowiedzi pod sfałszowane adresy, co powoduje, że pakiety te są rozpylane po całej przestrzeni adresowej Internetu. Można je wyłapywać *teleskopem pakietów* (ang. *packet telescope*), programem monitorującym nadchodzący ruch w ogłaszanych, ale nieużywanych sieciach.

Z efektem tym zetknęliśmy się w 1995 roku, gdy rozpoczęliśmy rozgłaszanie, wtedy nieużywanej, sieci AT&T 12.0.0.0/8 i rozpoczęliśmy obserwowanie nadchodzącego strumienia pakietów. W ciągu dnia odbieraliśmy z Internetu od 5 do 20 MB losowych pakietów. Niektóre z nich wypłynęły z miejsc, które używały sieci 12 na swoje potrzeby wewnętrzne. Inne były wynikiem różnego rodzaju błędów konfiguracyjnych. Jednak najciekawsze pakiety pochodziły od komputerów poddawanych różnym atakom fałszowania adresu nadawcy. Ludzie o niezbyt czystych intencjach wybrali sobie nieużywaną sieć AT&T na źródło fałszowanych pakietów, może jako żart, a może po to, by wskazać na „firmę telefoniczną". To, co zobaczyliśmy, było nadchodzącym zewsząd „krzykiem umierających komputerów".

Temat ten rozwinęli informatycy w pracy pod redakcją Moore'a [2001]. Obserwowali oni i analizowali ruch rozpraszania wstecznego (ang. *backscatter*) w celu zbadania rzeczywistego globalnego tempa i celów ataków. Rzadko udaje się zdobyć technikę dowodzącą powszechności ataków w skali globalnej. Poza zastosowaniami badawczymi, informacje te mają wartość komercyjną, gdyż wiele firm monitoruje swoich klientów w poszukiwaniu potencjalnych problemów, a teleskop pakietów jest świetnym detektorem wcześnie wykrywającym ataki typu DoS.

Posługując się siecią /8, wyłapaliśmy 1/256 losowo adresowanych pakietów w sieci. Znacznie mniejsze sieci, czyli i mniejsze teleskopy, wciąż mogą dać dobry obraz tego rodzaju ruchu — sieci /16 są z pewnością dostatecznie duże. Z wyliczeń wynika, że sieć /28 (zawierająca 16 komputerów) otrzymuje dziennie około sześciu tego typu pakietów.

Oczywiście, technika ta powoduje dalszy wyścig zbrojeń. Napastnicy mogą zacząć przestać używać adresów nadzorowanych sieci. Jednak gdy teleskopy pakietów zostaną umieszczone w różnych losowo wybranych małych sieciach, może być ciężko ujść uwadze sieciowych astronomów.

5.9. Roboty sieciowe

Programy zombi wykorzystywane do ataków DDoS to tylko wierzchołek góry lodowej. Wielu hakerów skonstruowało roboty sieciowe (ang. *botnets*, od *bots — robots —* roboty), czyli grupy botów, zombi, itp., które wykorzystują do swoich nieuczciwych poczynań.

Najbardziej znanym jest naturalnie, opisany wcześniej, atak DDoS. Botnety są również używane do rozproszonego wyszukiwania słabych miejsc. Dlaczego wykorzystywać do tego celu swój komputer, skoro można wykorzystać setki komputerów innych osób? Marcus Leech snuł domysły na temat wykorzystania robaków do łamania haseł lub rozproszonego rozszyfrowywania [Leech, 2002] internetowej wersji Chińskiej Loterii [Quisquater i Desmedt, 1991]. Kto wie, czy takie rzeczy już nie mają miejsca?

Botnety tworzone są za pomocą tradycyjnych środków: koni trojańskich, a w szczególności robaków. Jak na ironię, jednym z popularniejszych koni trojańskich jest konstruktor robotów. Osoba, która go używa, myśli, że buduje własną sieć botów, a tak na prawdę jej bot (i jej komputer) staje się częścią czyjegoś botnetu.

Wykorzystywanie robaków do budowy botnetów — jednym z tego przykładów[5] jest *slapper* — może być ryzykowne z powodu potencjalnej możliwości lawinowego ich rozprzestrzeniania się [Staniford *et al.*, 2002]. Niektóre robaki nawet poszukują innych, zainstalowanych wcześniej robaków oraz przejmują boty należące do kogoś innego. Program główny porozumiewa się z robotami na wiele różnych sposobów. Jednym z faworyzowanych sposobów komunikacji są kanały IRC, już przystosowane do masowej komunikacji. Nie trzeba zatem tworzyć dodatkowej infrastruktury komunikacyjnej. Polecenia są oczywiście zaszyfrowane. Wśród poleceń są też takie, które każą robotowi uaktualnić swój kod. W końcu jaką korzyść przynoszą nieaktualne roboty?

5.10. Ataki aktywne

W literaturze kryptograficznej opisywane są dwa rodzaje napastników. Pierwszy jest przeciwnikiem pasywnym, podsłuchującym całą komunikację sieciową, a jego zadanie polega na zebraniu tak wielu tajnych informacji, jak to możliwe. Drugi jest aktywnym intruzem, który wedle własnego uznania zmienia treść komunikatów, umieszcza w strumieniach danych swoje własne pakiety lub usuwa komunikaty. Wiele z prac teoretycznych bazuje na modelu systemu w postaci sieci gwiaździstej z napastnikiem w jej środku. W takim modelu każdy komunikat (pakiet) wędruje do napastnika, który może go odnotować, zmodyfikować, powielić, porzucić, itd. Napastnik może też tworzyć komunikaty i wysyłać je tak, by wyglądały, że pochodzą od kogoś innego.

Napastnik musi się ulokować w sieci między komunikującymi się ofiarami w ten sposób, by mógł obserwować przechodzące pakiety. Pierwszy publicznie opisany aktywny atak na protokół TCP, który opierał się na odgadywaniu numeru sekwencyjnego, został opisany w 1985 roku przez Morrisa [1985]. W tamtych czasach ataki te były interesujące z teoretycznego punktu widzenia, dziś istnieją programy, które przeprowadzają takie ataki automatycznie. Programy, takie jak *Hunt*, *Juggernaut* czy *IP-Watcher*, służą do przechwytywania połączeń TCP.

[5] CERT Advisory CA-2002-27 z 14 września 2002 — *przyp. aut.*

Niektóre ataki aktywne muszą pozbawić komunikacji jedną z upoważnionych stron (często za pomocą ataku odmowy usług) i podawać się za nią drugiej stronie. Aktywny atak na obie strony istniejącego połączenia TCP jest trudniejszy, ale również został już wykonany [Joncheray, 1995]. Powodem tego utrudnienia jest to, że obie strony połączenia TCP przechowują jego stan, który zmienia się z każdym wysłanym lub odebranym komunikatem. Ataki tego typu są wykrywane przez monitor sieciowy, gdyż nagle w sieci pojawia się dużo dodatkowych potwierdzeń i powtórzeń pakietów, ale mogą też przedostać się niezauważone przez użytkownika.

Nowsze narzędzia ataku fałszują komunikację ARP w celu ulokowania się w środku. Gdy komunikaty konsoli ostrzegają o wprowadzaniu zmian w informacjach zebranych przez protokół ARP, nie należy ich ignorować…

Kryptografia użyta w wyższych warstwach może posłużyć do odparcia ataku prowadzanego w warstwie transportowej, ale jedyną odpowiedzią na niego może być zerwanie połączenia. Atakom przejmowania połączeń mogą zapobiec techniki kryptograficzne warstwy łącza danych lub warstwy sieciowej, takie jak na przykład IPsec. Oczywiście, ataki aktywne mogą być przeprowadzane również w warstwie aplikacji. Przykładem tego jest atak typu *człowiek w środku* (ang. *man in the middle*) przeprowadzony na protokół uzgadniania klucza Diffie-Hellmana. Ataki aktywne na warstwę polityczną wykraczają poza zakres tematyczny tej książki.