helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

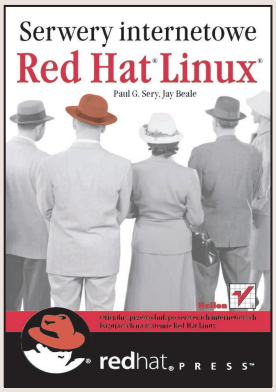
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

Serwery internetowe Red Hat Linux



Autorzy: Paul G. Sery, Jay Beale
Tłumaczenie: Rafał Szpoton (rozdz. 1 - 9, 19, dod. A - C), Marek Pętlicki (rozdz. 10 - 18)
ISBN: 83-7361-415-X
Tytuł oryginału: Red Hat Linux Internet Server
Format: B5, stron: 536

Niniejsza książka została zrecenzowana oraz zaakceptowana przez zespół ekspertów z firmy Red Hat. Zawiera ona informacje niezbędne do poprawnej konfiguracji serwerów internetowych działających pod kontrolą Linuksa, jak również zarządzania nimi.

Eksperci od systemu Red Hat Linux: Paul Sery oraz Jay Beale rozpoczynają od podstaw – istotnych informacji na temat zagadnień sieciowych, połączeń z siecią Internet, zapór sieciowych. Następnie w sposób bardziej szczegółowy przedstawiają sposób konfigurowania usług sieciowych oraz internetowych, począwszy od połączenia z bazą danych, tworzenia połączeń bezpiecznych i udostępniania mediów strumieniowych. W dalszej kolejności zajmują się konfiguracją serwerów nazwy domenowej, usług poczty elektronicznej, serwerów FTP oraz Samba. W kolejnych rozdziałach autorzy poświęcają wiele miejsca zarządzaniu serwerami, jak również umieszczają porady dotyczące utrzymywania zabezpieczeń sieci.

Ten autoryzowany przewodnik okaże się nieocenioną pomocą podczas tworzenia bezpiecznego, wydajnego serwera internetowego opartego na systemie Red Hat. Autorzy podają sprawdzone sposoby implementacji serwerów internetowych opartych na systemie Red Hat Linux.

- Konfiguracja sieci komputerowej w jednej z dwóch przedstawionych topologii
- Ustanowienie połączenia z siecią Internet za pomocą modemu kablowego lub DSL
- Tworzenie zapory sieciowej dokonującej filtrowania pakietów IP z uwzględnieniem ich stanu
- Konfiguracja serwera WWW Apache 2 oraz serwera strumieniowych danych audio
- Połączenie z bazą danych SQL z poziomu serwera WWW oraz tworzenie bezpiecznych połączeń SSL
- Tworzenie serwerów DNS, SMTP, FTP oraz Samba
- Automatyzacja tworzenia kopii bezpieczeństwa poprzez sieć komputerową
- Zabezpieczanie serwerów i wykrywanie włamań

O autorach:

Paul G. Sery pracuje dla Narodowego laboratorium Sandia w Albuquerque, w stanie Nowy Meksyk, w którym zajmuje się projektowaniem sieci oraz zarządzaniem nimi.

Jay Beale jest założycielem oraz przewodniczącym firmy konsultingowej JJB Security Consulting, jak również głównym programistą projektu Bastille Linux, który ma być używany do lepszego zabezpieczania systemów Linux oraz HP-UX. Często wykłada na konferencjach poświęconych bezpieczeństwu systemów.

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

Spis treści

O Autorach **13**

Przedmowa **15**

Część I Tworzenie sieci komputerowej w oparciu o system Linux **19**

Rozdział 1. Przykłady sieci komputerowych **21**

Przykłady sieci komputerowych 22
- Bezpośrednie połączenie z siecią internetową (DCI) 22
- Połączenie z siecią internetową za pośrednictwem strefy zdemilitaryzowanej (DMZI) 23

Opis funkcji serwera 25
- Podstawy konfiguracji sieci DCI 26
- Podstawy konfiguracji DMZI 28

Dodawanie podsieci do obu przykładowych sieci komputerowych 29
Podsumowanie 30

Rozdział 2. Konfiguracja usług sieciowych w systemie Red Hat **33**

Podstawy protokołu internetowego IP 34
- Usługi sieciowe a model OSI 34
- Podstawy protokołów warstwy transportowej 35

Podstawy protokołów warstwy sieci (przekazywanie pakietów IP) 36
Działanie protokołów warstwy łącza (ramki w sieci Ethernet) 37
Wygląd sieci opartej na systemie Red Hat Linux 38
- Ważne pliki konfiguracyjne sieci 38
- Ważne aplikacje oraz skrypty 42
- arp/rarp 42
- ifconfig 43
- netstat 44
- nmap 44
- ping 44
- redhat-config-network-druid/ redhat-config-network-cmd 45
- route 45
- sysctl 46
- tcpdump 47
- modprobe 47

Konfiguracja jednej lub większej liczby kart sieciowych ... 48
Przykład 1: Konfiguracja pojedynczego interfejsu sieciowego ... 48
Przykład 2: Konfiguracja dwóch interfejsów sieciowych (dwie podsieci) ... 52
Przykład 3: Konfiguracja routera w oparciu o system Red Hat Linux ... 55
Podsumowanie ... 59

Rozdział 3. Nawiązywanie połączenia z siecią Internet przy użyciu łącza DSL ... 61
Podstawy technologii DSL ... 62
Terminologia stosowana w technologii DSL ... 66
Ogólne definicje ... 66
Rodzaje usług DSL ... 70
Tworzenie połączenia z siecią Internet przy użyciu łącza DSL ... 71
Uzyskiwanie połączenia internetowego DSL ... 71
Fizyczne podłączanie modemu DSL ... 71
Konfiguracja wyposażenia udostępnianego przez dostawcę usługi DSL ... 73
Podstawowe wskazówki przydatne podczas rozwiązywania problemów ... 79
Podsumowanie ... 83

Rozdział 4. Tworzenie systemu zapory sieciowej ... 85
Podstawy systemów zapór sieciowych ... 85
Zapory sieciowe filtrujące pakiety ... 86
Zapory pośredniczące ... 87
Zapory hybrydowe ... 88
Podstawy translacji adresów (NAT) ... 88
Wyjaśnienie przepływu pakietów w sieci Internet ... 89
Podstawy stanowych filtrów pakietów (Netfilter/iptables) ... 91
Tworzenie systemu zapory sieciowej ... 97
Ochrona sieci za pomocą prostego zestawu reguł ... 97
Zacieśnianie zapory sieciowej przy użyciu łańcuchów definiowanych przez użytkownika ... 104
Wpuszczanie z zewnątrz połączeń SSH ... 109
Konfiguracja serwera SSH ... 112
Modyfikacja serwera zapory sieciowej (atlas) w celu umożliwienia korzystania z usługi SSH ... 114
Zarządzanie systemem zapory sieciowej ... 116
Podsumowanie ... 117

Rozdział 5. Rozwiązywanie prostych problemów ... 119
Rozwiązywanie problemów przy użyciu drzewa decyzyjnego ... 120
Rozwiązywanie problemów sieciowych w systemie Red Hat Linux ... 121
Czy zostało włączone zasilanie? ... 121
Czy nie została uszkodzona fizyczna instalacja sieci komputerowej? ... 122
Czy przełącznik sieciowy lub koncentrator jest skonfigurowany w sposób poprawny? ... 122
Czy interfejs sieciowy został skonfigurowany w sposób prawidłowy? ... 122
Rozwiązywanie problemów z połączeniem DSL ... 129
Sprawdzanie konfiguracji modemu DSL (routera) po stronie użytkownika ... 130
Sprawdzanie konfiguracji dostawcy usługi DSL ... 131
Sprawdzanie konfiguracji ISP ... 132
Rozwiązywanie problemów z bramką i zaporą sieciową ... 132
Sprawdzanie konfiguracji sieciowej w systemie Red Hat Linux ... 133
Sprawdzanie tras bramki oraz przekazywania pakietów IP ... 134
Sprawdzanie skryptów zapory sieciowej ... 136

Sprawdzanie modułów jądra oraz znaczników ... 137
Wykorzystywanie narzędzi podsłuchujących ruch w sieci ... 137
Dodatkowe informacje ... 139
Podsumowanie ... 139

Część II Tworzenie serwera WWW ... 141

Rozdział 6. Prosta konfiguracja serwera Apache ... 143
Podstawy języka HTML oraz protokołu HTTP ... 143
Protokół HTTP ... 143
Język HTML (HTML nie jest protokołem) ... 144
Serwer WWW Apache ... 148
Dyrektywy konfiguracyjne serwera Apache ... 149
Plik konfiguracyjny serwera Apache ... 149
Tworzenie prostej witryny WWW ... 158
Instalacja serwera Apache ... 158
Konfiguracja serwera Apache ... 159
Kontrolowanie działania serwera Apache ... 162
Używanie serwera Apache ... 163
Kontrola dostępu do serwera WWW za pomocą plików .htaccess oraz .htpasswd ... 163
Uruchamianie wirtualnego serwera WWW ... 164
Stosowanie protokołu SSL z serwerem Apache ... 167
Instalacja pakietu mod_ssl ... 168
Negocjacja SSL oraz certyfikaty ... 168
Centra autoryzacji ... 169
Konfiguracja serwera Apache w celu wykorzystywania protokołu HTTP z SSL ... 169
Tworzenie certyfikatów podpisanych przez CA ... 170
Rozwiązywanie problemów ... 171
Sprawdzanie systemowych plików dziennika systemu Linux ... 172
Sprawdzanie plików dziennika serwera Apache ... 172
Połączenie lokalne ... 173
Sprawdzanie konfiguracji serwera Apache ... 173
Jeśli mamy taką możliwość, należy użyć uproszczonego pliku httpd.conf ... 173
Stopniowe dodawanie nowych dyrektyw ... 174
Podsumowanie ... 174

Rozdział 7. Tworzenie połączenia z bazą danych ... 175
Podstawy języka SQL ... 175
Instalacja oraz konfiguracja bazy MySQL ... 177
Dostęp do serwera SQL ... 178
Tworzenie bazy danych ... 179
Wykorzystywanie bazy danych MySQL ... 181
Wymiana danych z serwerem SQL przy użyciu skryptów ... 182
Wyświetlanie danych z bazy danych MySQL ... 183
Wstawianie danych do bazy danych MySQL ... 186
Modyfikacja danych w bazie danych MySQL ... 187
Wykorzystywanie skryptu CGI w celu dostępu do bazy danych SQL za pomocą przeglądarki ... 190
Zabezpieczanie bazy danych MySQL ... 195
Podsumowanie ... 196

Rozdział 8. Tworzenie prostego serwera multimedialnego ... 197
Wprowadzenie do technologii strumieniowej ... 197
Fundacja Xiph.org ... 198

Udostępnianie strumieni danych dźwiękowych przy użyciu serwera Icecast....200
Instalacja oraz konfiguracja serwera Icecast....200
Instalacja oraz konfiguracja źródła Ices....202
Udostępnianie strumieni MP3....203
Udostępniane strumieni Ogg Vorbis....205
Rozwiązywanie problemów....218
Podsumowanie....220

Część III Udostępnianie prostych usług internetowych....221

Rozdział 9. Tworzenie serwera DNS....223
Podstawy systemu DNS....223
Domeny....224
Strefy....224
Autorytatywne serwery nazw....225
Podstawy procesu tłumaczenia nazwy przez klienta....226
Analiza prostego żądania przetłumaczenia nazwy....227
Rekordy zasobów....230
Stosowanie rekordu SOA....230
Definiowanie rekordów zasobów strefy....232
Instrukcje konfiguracyjne oraz parametry demona bind....232
Pliki konfiguracyjne /var/named/....234
Konfiguracja prostego serwera DNS....235
Konfiguracja nadrzędnego serwera nazw....236
Konfiguracja pomocniczego serwera nazw....240
Dodawanie środków bezpieczeństwa....241
Stosowanie list ACL....241
Stosowanie podpisów cyfrowych podczas wymiany stref....243
Uruchamianie demona named w środowisku chroot....244
Uruchamianie oraz zatrzymywanie działania serwera nazw....245
Tworzenie wielu plików stref....246
Konfiguracja serwera rozdzielonej domeny....248
Konfiguracja serwera nazw dla sieci prywatnej w konfiguracji rozdzielonej domeny....249
Konfiguracja serwera nazw dla sieci DMZ w konfiguracji domeny podzielonej....251
Rozwiązywanie problemów....252
named-checkzone....252
named-checkconf....252
dig....253
host....253
tcpdump....254
Podsumowanie....254

Rozdział 10. Konfiguracja serwera pocztowego SMTP....255
Trochę teorii na temat poczty elektronicznej....255
Kompatybilność....256
Wydajność....257
Wykorzystanie....257
Kolejki....257
Parametry konfiguracyjne....257
Składnia parametrów....258
Wartości bezpośrednie....258
Pliki....258
Bazy danych i tablice....258

Obsługa niechcianej poczty....259
Najważniejsze parametry z pliku main.cf....259
queue_directory....259
command_directory....259
daemon_directory....260
mail_owner....260
default_privs....260
myhostname....261
mydomain....261
myorigin....261
inet_interfaces....261
mydestination....262
local_recipient_maps....262
masquerade_domains....262
masquerade_exceptions....262
local_transport....263
alias_maps....263
alias_database....263
home_mailbox....263
mail_spool_directory....263
mailbox_command....264
mailbox_transport....264
fallback_transport....264
luser_relay....264
smtpd_recipient_limit....265
smtpd_timeout....265
mynetworks_style....265
mynetworks....266
allow_untrusted_routing....266
maps_rbl_domains....266
smtpd_client_restrictions....266
smtpd_sender_restrictions....267
smtpd_recipient_restrictions....267
smtpd_helo_required....267
smtpd_helo_restrictions....267
smtpd_delay_reject....268
strict_rfc821 _envelopes....268
header_checks....268
body_checks....268
message_size_limit....269
relay_domains....269
mynetworks....269
smtpd_banner....269
local_destination_concurrency_limit....270
default_destination_concurrency_limit....270
debug_peer_list....270
debug_peer_level....270
debugger_command....270
disable_vrfy_command....271
Konfiguracja podstawowych plików....271
Konfiguracja pliku master.cf....271
Konfiguracja pliku aliases....271
Konfiguracja pliku virtual....272
Konfiguracja pliku canonical....272
Konfiguracja pliku access....272

Wykorzystanie poleceń administracyjnych....273
Terminologia serwerów poczty elektronicznej....273
Programy typu Mail User Agent....274
Skład poczty....274
Programy typu Mail Transport Agent....274
Nagłówki poczty....275
Koperta....275
Przykładowe konfiguracje....276
Przykład 1: Wysyłanie poczty....276
Przykład 2: Przyjmowanie poczty dla różnych domen....276
Przykład 3: Wirtualne domeny w stylu systemu Postfix....277
Przekazywanie poczty między adresami wirtualnymi....278
Przykład 4: Weryfikacja ustawień DNS dla serwera poczty....280
Przykład 5: Przekazywanie całej poczty przez serwer centralny....287
Przykład 6: Konfiguracja serwera centralnego....288
Przykład 7: Minimalizacja ilości niechcianej poczty....288
Podstawy systemu spamassassin....290
Podsumowanie....291

Rozdział 11. FTP....293
Podstawy protokołu FTP....293
Washington University FTP (WU-FTPD)....294
Instalacja WU-FTPD....295
Plik konfiguracyjny /etc/xinetd.d/wu-ftpd....296
Plik ftpaccess....298
Plik ftpconversions....302
Konfiguracja serwera FTP trybu rzeczywistego....303
Konfigurowanie kont gościnnych....303
Konfiguracja kont anonimowych....305
Konfiguracja kont anonimowych....305
Konfiguracja możliwości anonimowego przesyłania plików na serwer....306
Rozwiązywanie problemów z serwerem WU-FTPD....308
Kontrola na poziomie ogólnym....309
Problemy z trybem gościnnym....311
Problemy z trybem anonimowym....313
Podsumowanie....314

Rozdział 12. Konfiguracja Samby....315
Wprowadzenie do Samby....315
Składnia pliku smb.conf....317
Parametry pliku smb.conf....317
Struktura pliku smb.conf....318
Przykłady konfiguracji Samby....319
Konfiguracja Samby w celu wykorzystania szyfrowanych haseł....319
Tworzenie zasobów Samby....323
Dostęp do macierzystych katalogów użytkowników....323
Uprawnienia Linuksa i Samby....324
Wykorzystanie makr Samby....330
Udostępnianie drukarek sieciowych za pomocą Linuksa i Samby....331
Narzędzie SWAT....335
Rozwiązywanie problemów....335
Podsumowanie....336

Część IV Zarządzanie serwerami linuksowymi....339

Rozdział 13. Automatyzacja sieciowych kopii zapasowych....341
Wprowadzenie do systemu AMANDA....342
Szczegóły systemu AMANDA....343
Części serwerowa i kliencka systemu AMANDA....343
Serwisy sieciowe....344
Pliki konfiguracyjne....345
Narzędzia systemu AMANDA....347
Wykorzystanie systemu AMANDA....348
Konfiguracja minimalistycznego systemu kopii zapasowych....349
Konfiguracja prostego systemu kopii zapasowych....352
Automatyzacja procesu wykonywania kopii zapasowych....357
Rozwiązywanie problemów....357
Podsumowanie....358

Rozdział 14. Zwiększanie niezawodności serwera....361
Lokalizacja słabych punktów....361
Wykorzystanie systemu plików ext3....363
Wykorzystanie macierzy RAID....365
RAID programowy....365
Implementacja programowej macierzy RAID....368
Tworzenie klastra o wysokiej dostępności....370
Mechanizm działania HA....371
Tryby przejęcia obsługi....371
Tworzenie prostego klastra wysokiej dostępności na Linuksie....372
Testowanie systemu Heartbeat....374
Podsumowanie....375

Część V Zwiększanie bezpieczeństwa....379

Rozdział 15. Podstawy bezpieczeństwa serwerów....381
Zrozumienie zagrożeń....381
Znaczenie poprawek systemowych....382
Identyfikacja napastników....384
Kategorie napastników....384
Przewidywanie ataków....386
Sposoby obrony....389
Podsumowanie....396

Rozdział 16. Podstawy bezpiecznej administracji systemów....397
Administrator systemu a administrator bezpieczeństwa....397
Automatyczna aktualizacja, łaty i zagrożenia....398
Środowisko produkcyjne i rozwojowe....400
Automatyczne instalowanie poprawek....402
Minimalizacja, standaryzacja i upraszczanie....404
Zrozumieć minimalizację....404
Trzy cnoty główne....407
Monitorowanie i bezpieczna zdalna administracja....407
Centralny system zarządzania....409
Monitorowanie stanu....410
Podsumowanie....411

Rozdział 17. Wzmacnianie systemu 413
Zrozumieć proces wzmacniania systemu 413
Ręczne wzmacnianie systemu 415
Blokowanie lub usuwanie niepotrzebnych programów 415
Powtórka z procesu uruchamiania Linuksa 417
Kontynuacja audytu demona sieciowego 422
Automatyczne wzmacnianie systemu za pomocą Bastille Linux 435
Moduł zabezpieczeń kont użytkowników 435
Moduł bezpieczeństwa procesu uruchamiania systemu 436
Konfiguracja ustawień mechanizmu PAM 436
Moduł deaktywacji demonów 436
Moduł blokujący dostęp do narzędzi 437
Moduł uprawnień do plików 437
Moduł dzienników 437
Moduł wydruku 438
Moduł zabezpieczania inetd lub xinetd 438
Moduł zabezpieczenia katalogu tymczasowego 438
Moduł Apache 439
Moduł BIND (DNS) 439
Moduł FTP 439
Moduł sendmail 440
Moduł zapory sieciowej 440
Moduł detekcji skanowania portów 440
Podsumowanie 440

Rozdział 18. Proste systemy wykrywania włamań 443
Sieciowe i systemowe wykrywanie włamań 443
Określenie zakresu odpowiedzialności 444
Odpowiedzialność administratora 444
Odpowiedzialność oznacza władzę 446
Mechanizmy sieciowe 447
Pakiety 447
Adresy IP 449
Porty 451
ICMP 454
UDP 454
TCP 454
Rozpoznania i zagrożenia 456
Projektowanie defensywnej konfiguracji sieci 459
Filtrowanie na routerach oraz zapory sieciowe 459
Filtrowanie ingress i egress 460
DMZ 460
Hosty bastionowe 460
Dedykowane serwery 461
Projektowanie strategii wykrywania włamań 461
Ustalanie reguł 462
Detektory włamań nie wykrywają włamań 463
Pozytywne specyfikacje są złe 463
Negatywne specyfikacje nie są lepsze 463
Heurystyczne wykrywanie anomalii — Święty Graal 464
Przykładowy system NIDS — Snort 464
Opis i historia 464
Przegląd funkcji systemu Snort (wybrane opcje wiersza poleceń) 465
Podsumowanie 467

Rozdział 19. Monitorowanie plików dziennika oraz reagowanie na zdarzenia 469
Odczytywanie systemowych plików dziennika 469
Rejestrowanie danych w plikach dziennika 470
Rejestrowanie danych w plikach dziennika aplikacji 471
Rejestrowanie danych w plikach dziennika systemu 472
Rejestrowanie danych w hybrydowych plikach dziennika 473
Syslog 474
Wykonywanie akcji wybranych przez selektor 475
Pola selektorów, jednostki oraz priorytety 475
Miejsca docelowe 477
Wykorzystywanie plików dziennika 478
Analiza bezpośrednia 479
Narzędzia przetwarzające pliki dziennika 479
egrep 480
Tworzenie raportów 482
Narzędzie monitorujące pliki dziennika 485
Zarządzanie odpowiedziami na zdarzenia 486
Najgorsza jest panika 486
Przewidywanie oznacza politykę bezpieczeństwa 487
Podsumowanie 487

Dodatki 489

Dodatek A Konfiguracja komutowanego połączenia typu dial-up z siecią Internet ... 491
Lokalizacja modemu 491
Nawiązywanie połączenia z siecią Internet 493

Dodatek B Automatyzacja konfiguracji serwera 499
Instalacja systemu przy użyciu programu Kickstart 500
Tworzenie pliku konfiguracyjnego programu Kickstart 500
Tworzenie dyskietki startowej programu Kickstart 501
Instalacja przy użyciu programu Kickstart 503
Tworzenie własnych pakietów RPM 503
Wygląd pakietu RPM 504
Przykładowy pakiet RPM 504

Dodatek C Stosowanie usługi DHCP 507
Instalacja dhcpd 507
Konfiguracja stacji roboczej działającej pod kontrolą systemu Red Hat Linux w celu korzystania z usługi DHCP 508
Konfiguracja komputera działającego pod kontrolą systemu Windows w charakterze klienta DHCP 509

Skorowidz 511

Rozdział 18.

Proste systemy wykrywania włamań

W niniejszym rozdziale zajmiemy się systemami wykrywania włamań (*Intrusion Detection System — IDS*). System IDS nie wie, kim jest intruz. Taki system śledzi zdarzenia i warunki związane z wrogą aktywnością w komputerze lub sieci. Ktoś — a konkretnie administrator — musi określić, które sytuacje oznaczają włamanie i upewnić się, że IDS będzie na nie odpowiednio reagować.

W rzeczywistości większość systemów IDS ma określony zestaw domyślnych warunków wskazujących, że w systemie dzieje się coś niedobrego, lecz każdego dnia pojawiają się nowe sytuacje tego typu, więc ktoś musi uzupełniać ustawienia IDS, tak aby reagował na takie „niestandardowe" przypadki. Jak się wkrótce przekonamy, administrator znajduje sytuacje specyficzne dla konkretnej konfiguracji, które chciałby uwzględnić w reakcjach systemu wykrywania, a które nie były przewidziane w standardowej konfiguracji udostępnionej przez dostawcę systemu IDS.

Sieciowe i systemowe wykrywanie włamań

Wielu obcych i złych ludzi przygląda się ciągle naszym komputerom. Być może spotkałeś się już z ich śladami wyzierającymi z katalogów, w których wprowadzili własne poprawki i podmienili narzędzia systemowe na własne, które bardziej im się podobały.

Biznes alarmów antywłamaniowych kwitnie najbardziej w okresach, kiedy włamywacze są bardziej aktywni. Z tego samego powodu wysoką popularnością cieszą się „alarmy antywłamaniowe" sieci komputerowych, czyli systemy IDS. Wielu sprzedawców wmusza je tak samo, jak robią to z zaporami sieciowymi — sugerują jakoby były cudownym środkiem rozwiązującym wszelkie problemy z intruzami. Jeśli wierzysz fachowcom od marketingu, powtórzymy to, co mówiliśmy już kilkakrotnie: nie ma cudownych środków. Naprawdę. Zaufaj nam.

Nie chcemy jednak deprecjonować znaczenia zapór sieciowych czy systemów IDS. Zarówno zapory sieciowe jak, i IDS są potrzebne pod warunkiem, że zostały właściwie zaprojektowane. Nie uzyskamy większych korzyści z ich stosowania, jeśli ich implementacja nie będzie poprzedzona dokładną analizą konkretnej sieci i systemów. W tym rozdziale omówimy ogólne koncepcje związane z systemami IDS, a następnie przedstawimy bardzo ciekawy i darmowy system sieciowego wykrywania włamań (NIDS) o nazwie Snort. Nawiasem mówiąc, skrót *NIDS* oznacza *Network Intrusion Detection System*, czyli system sieciowego wykrywania włamań, natomiast *HIDS* jest skrótem od *Host Intrusion Detection System*, czyli systemu wykrywania włamań na poziomie systemu (systemowy).

Sieciowe systemy IDS śledzą zdarzenia w sieci w oczekiwaniu na ataki. Systemowe IDS działają w monitorowanym systemie w oczekiwaniu na nieautoryzowany dostęp, a nawet na nietypowe zachowanie systemu i użytkowników. Zaletą podejścia sieciowego jest lepsza skalowalność i o wiele większa prostota rozwiązania. W rezultacie w większości przypadków pojęcie IDS jest utożsamiane z podejściem sieciowym do tego zagadnienia, czyli z NIDS.

Teoretycznie podejście systemowe cechuje się większą dokładnością, z mniejszym prawdopodobieństwem fałszywych alarmów, lecz dość trudno o dobrą implementację systemu HIDS opartego o analizę nietypowych zachowań. Najpopularniejsza implementacja systemu HIDS polega na sprawdzaniu integralności plików systemowych. Program tego typu monitoruje najważniejsze pliki w systemie pod kątem wprowadzanych zmian. Podejście to opiera się o stwierdzenie, że włamywacz w celu zwiększenia swoich uprawnień lub zachowania dostępu do systemu na stałe z reguły stara się zmodyfikować ważne konfiguracje w systemie.

W kolejnym podrozdziale zapoznamy się z koncepcją NIDS jako częścią planowania systemu bezpieczeństwa. Rozpoczniemy od problemu określenia roli, jaką chcemy powierzyć podsystemowi NIDS w całym systemie.

Określenie zakresu odpowiedzialności

Zdecydowaliśmy się zatem zaimplementować NIDS w naszej sieci. Jak wspomnieliśmy, wiąże się to z koniecznością przeprowadzenia wstępnej analizy i planowania. Zanim zajmiemy się więc tą najbardziej ekscytującą częścią zadania, jaką niewątpliwie są ataki sieciowe, musimy odpowiedzieć sobie na kilka ważnych pytań tak oczywistych, że często zapominamy sobie na nie odpowiedzieć. Bez odpowiedzi na nie nie ma sensu dalsze działanie w tym kierunku. Pytania te przeanalizujemy w kolejnych podrozdziałach.

Odpowiedzialność administratora

Ten rozdział książki zajmuje się bezpieczeństwem, rozpatrzymy zatem odpowiedzialność administratora w dziedzinie bezpieczeństwa.

Im więcej zastanawiamy się nad komputerem — indywidualnie lub jako częścią sieci — tym bardziej dochodzimy do przekonania, że trzeba go chronić lub pilnować, a raczej jedno i drugie! Podczas pierwszej sesji kontroli ruchu w sieci administratorzy niezmiennie dają się zaskoczyć ilością odbywających się transakcji. Takie samo odczucie towarzyszy im podczas pierwszej kontroli dzienników na scentralizowanym serwerze dzienników systemu sieciowego. Uświadomienie sobie implikacji każdego bitu ruchu sieciowego i każdego wpisu w dziennikach stanowi tę łatwiejszą części większego zadania. Prawdziwa praca zaczyna się, gdy musimy każdą z tych części przeanalizować w powiązaniu z innymi zdarzeniami, szczególnie wtedy gdy wciąż napływają nowe informacje i pojawiają się nowe potencjalne zagrożenia systemu.

Jeśli chcemy utrzymać kontrolę, od samego początku musimy narzucić sobie ograniczenia. Musimy określić ramy odpowiedzialności i możliwości projektowanego systemu bezpieczeństwa, zanim zaczniemy zastanawiać się, w jaki sposób ustawić poszczególne jego elementy. We wszystkich tych przypadkach musimy zastanowić się, co musimy chronić, i nakreślić zakres naszych wymagań dotyczących bezpieczeństwa.

- **Zarządzanie systemem dla pracodawcy**. Jeśli dla kogoś pracujesz, musisz wiedzieć, czego się od Ciebie oczekuje. Odpowiedź na tę kwestię musi być ujęta na piśmie, jak w umowie. Co mam chronić? Te zagadnienia formułują zakres odpowiedzialności. Każda strategia NIDS (lub pokrewna) musi zawierać wszystkie elementy bezpieczeństwa w określonym zakresie odpowiedzialności. Jeśli określona strategia przynosi korzyść innym, to doskonale, lecz przede wszystkim należy skupić się na realizacji własnych zadań, jeśli chcemy osiągnąć jakiekolwiek rezultaty.
- **Zarządzanie systemem we własnej firmie**. Jeśli system, którym zarządzasz, służy Twoim własnym interesom, musisz określić swoje oczekiwania w stosunku do niego. Jeśli Twoja firma prowadzi serwisy WWW, nie ma potrzeby udostępniania serwera IRC (w każdym razie nie na tej samej maszynie!). Po określeniu wymagań biznesowych wiemy już, co musimy chronić. Po prostu chronimy możliwości spokojnego prowadzenia interesów. Ta zasada przekłada się bezpośrednio na ochronę określonych systemów, a nawet określonych funkcji systemów.
- **Zarządzanie systemem na użytek prywatny**. Można również zarządzać systemem na własny użytek lub też w celu udostępnienia go grupce przyjaciół. Dla zabawy. Wtedy chcesz chronić ten system dla siebie i swoich przyjaciół, aby kontynuować zabawę.

Oczywiście chcemy śledzić wszystkie zagadnienia dotyczące serwerów (lub przynajmniej dotyczące świadczonych przez nie usług). Chcemy również śledzić inne zagadnienia, na przykład tylko ruch sieciowy skierowany do serwerów w naszej sieci. Jeśli jednak przytrafią się podejrzane pakiety skierowane do serwera lub kilku z nich, nie mamy pewności, czy to były skany w poszukiwaniu słabych punktów systemu, czy też śmieci w wyniku jakiegoś błędu sprzętowego urządzeń sieciowych w drugim końcu świata. Jeśli jednak monitorujemy cały ruch sieciowy i widzimy, że takie same pakiety trafiły do większej liczby maszyn w sieci w kolejności zgodnej z numeracją adresów, wiemy już, że mamy do czynienia ze skanem. Błędy sprzętowe nie charakteryzują się takim uporządkowaniem.

Gdy określimy zakres ochrony, musimy ustalić zakres nadzoru, aby uzyskać informacje wystarczające do zapewnienia tej ochrony. Należy określić zakres monitoringu sieciowego.

Zakres monitoringu mógłby z powodzeniem obejmować cały Internet i każdy komputer w Internecie, jak również podsystemy energetyczne, do których podłączony jest nasz system. Oczywiście nie mamy dostępu do tych informacji, dlatego musimy skoncentrować się na kontroli tych obszarów, które możemy kontrolować. Do tej grupy należy cały ruch przechodzący przez monitorowany system (niezależnie od tego, czy ruch jest skierowany do tego systemu). Mamy dostęp do wszystkich dzienników systemowych, które może wygenerować system (niestety nie wszystkie procesy, które są w stanie wygenerować informacje do dziennika, mają domyślnie aktywną tę funkcję). Czy posiadane informacje są wartościowe? Prawdopodobnie. Czy są wystarczające? Na pewno nie. Informacji nigdy nie jest wystarczająco dużo. Czy posiadana ilość informacji jest rozsądna z punktu widzenia potrzeb związanych z wykonywanymi przez nas zadaniami? To zależy od własnej oceny. Nie możemy na nikim wymusić zakresu prowadzonego monitoringu, każdy musi sam to określić. My możemy jedynie powiedzieć, że zakres monitoringu powinien odpowiadać zakresowi wymagań.

Odpowiedzialność oznacza władzę

W poprzednim podrozdziale milcząco założyliśmy, że administrator (czytelnik) jest w stanie podjąć odpowiednie środki bezpieczeństwa na wszystkich serwerach, którymi zarządza. Innymi słowy, nikt nie będzie pociągał go do odpowiedzialności za zagadnienia wybiegające poza zakres jego kontroli. W idealnym świecie nie byłoby w ogóle potrzeby podkreślania tego oczywistego faktu. W rzeczywistym jednak uważamy, że jest to zagadnienie, które warto wyraźnie podkreślić. Jeśli administrator uważa, że pewne zagadnienia są kluczowe w celu realizacji funkcji bezpieczeństwa, lecz nie ma nad nimi kontroli, powinien natychmiast podjąć ten temat na piśmie.

Notatka służbowa stanowi doskonałe narzędzie formalne, zwykle wykorzystywane w celu wyjaśnienia zagadnień. W naszym przypadku może być z powodzeniem użyta jako uprzejme zwrócenie uwagi na to, że ktoś podjął błędne decyzje. Oto przykład takiej notatki:

Od: Ktoś

Do: Szef i Szef Szefa

Data: 1 marca 2003r.

Temat: Hasło konta root na serwerze baz danych

No to co Ty właściwie robisz?

Jeśli z umowy lub zakresu obowiązków nie można jasno wywnioskować zakresu odpowiedzialności, należy wyegzekwować taki dokument na piśmie. Istnieje tendencja do obwiniania administratorów systemów za wszelkiego rodzaju problemy. Tym bardziej dotyczy to osób odpowiedzialnych za bezpieczeństwo systemów! Należy zatem jasno zdefiniować odpowiedzialność i wszystkie strony muszą potwierdzić tę umowę na piśmie.

Niniejsza notatka ma za zadanie udokumentować wyniki spotkania z 28 lutego 2003r. dotyczącego zagadnień bezpieczeństwa głównego serwera bazy danych. Po przeanalizowaniu zaleceń szefa systemu bezpieczeństwa (zob. poprzednia notatka z dnia 30 listopada 2002r.) dotyczącego wymogu wykorzystania haseł na koncie root służby wykorzystujące ten serwer uznały, iż spowoduje to nadmierne komplikacje procesu uwierzytelniania użytkowników przewyższające zalety uzyskane z podniesienia poziomu bezpieczeństwa.

Jak widać, notatka ta odwołuje się do poprzedniej notatki. Gdy administrator jest osobą zatrudnioną w firmie, pozostawianie śladu na papierze na temat wszelkich zdarzeń jest kluczowe dla bezpieczeństwa własnej pracy. Jeśli administrator pracuje na własną rękę, również powinien to robić, choćby po to, by później prześledzić zdarzenia i upewnić się, że właściciel pamięta o wszystkim, co powiedział.

Przed dalszą analizą przygotowującą do wdrożenia mechanizmu NIDS prześledźmy mechanizmy sieciowe w Internecie. Czytelnicy dobrze zaznajomieniu z terminologią sieci TCP/IP mogą pominąć te rozważania.

Mechanizmy sieciowe

Niniejszy podrozdział omawia zagadnienia związane z systemami wykrywania włamań, szczególnie sieciowymi; skupia się na konkretnym rozwiązaniu o nazwie Snort. Uzgodniliśmy, że do obowiązków administratora należy określenie zakresu kontroli zdarzeń w sieci w odniesieniu do specyfiki danej instalacji. Czytelnicy, którzy posiadają dobrą orientację w zakresie zagadnień TCP/IP oraz rodzajów ataków i skanowania sieci, mogą potraktować ten podrozdział jak podsumowanie. Pozostali znajdą w nim skromne wprowadzenie do grubych książek traktujących o tych tematach, których poznanie jest kluczowe w celu dalszej i skutecznej pracy z mechanizmami typu NIDS.

TCP/IP jest rodziną standardowych protokołów sieciowych wykorzystywanych w Internecie (w rzeczywistości *IP* jest skrótem od *Internet Protocol*). Protokół sieciowy można traktować jak język, którym porozumiewają się urządzenia sieciowe podczas przesyłania komunikatów przy użyciu kabli, światłowodów czy fal radiowych. W sieci mogą być przesyłane również i inne rodzaje ruchu, szczególnie jeśli wykorzystywany w niej jest system Microsoft Windows. Innych protokołów używa również sieć Novell. Nie będziemy jednak wnikać w szczegóły. Zajmiemy się rodziną protokołów TCP/IP, ponieważ to one obsługują komunikację w Internecie.

Pakiety

Informacje przesyłane w sieci są podzielone na kawałki, nie są przesyłane ciągłym strumieniem danych. Dzięki temu różne transakcje mogą regularnie otrzymywać fragmenty czasu w sieci. Najpierw przechodzi jeden z pakietów nadawcy A, następnie jeden od nadawcy B, następnie od C, potem znów od A. Właściwie większość aktywności odbywa się w trakcie pisania na klawiaturze czy przeglądania stron WWW.

Dane są przesyłane w porcjach opakowanych z obydwu stron dodatkowymi informacjami umożliwiającymi nawigację i dostarczanie ich w Internecie. Typ pakietu uzależniony jest od przeznaczenia przesyłanych danych oraz wymaganego typu transmisji.

System IDS, taki jak Snort, przegląda informację na temat pakietów, a także dane w nich zawarte. W rzeczywistości z samego pakietu można wyciągnąć mnóstwo informacji! Większość zapór sieciowych pracuje wyłącznie w oparciu o informacje na temat pakietu. Wiele decyzji podejmowanych przez te narzędzia polega wyłącznie na niewłaściwym wyglądzie „opakowania", niezależnie od zawartości pakietu.

Kapsułkowanie

Przyglądając się bliżej opakowaniu, stwierdzimy, że dane przesyłane w sieci raczej nie są zamykane w jednym pakiecie, lecz z reguły znajdują się w pakiecie, który znajduje się w innym pakiecie, który znajduje się w kolejnym — średnio wykorzystywane są cztery warstwy takiego „opakowania". Takie warstwowe pakiety można sobie wyobrazić jako koperty włożone jedna w drugą, gdy każda z nich zawiera pewne informacje częściowe, niezbędne w celu dostarczenia zawartych w nich danych. Taka informacja w zagnieżdżonych warstwach jest częścią strategii komunikacyjnej znanej pod nazwą kapsułkowania (*encapsulation*) lub izolacji. Taka strategia komunikacyjna jest niezbędna, ponieważ trasa przesyłania danych jest skomplikowana, a my na każdym etapie podróży nie chcemy zawierać w danych pakietu więcej informacji, niż to niezbędne. Pamiętajmy, że trasa pakietu prowadzi od procesu na jednej maszynie w konkretnej sieci, do drugiego procesu, najczęściej na innej maszynie, niejednokrotnie w zupełnie innej sieci.

Pakiet musi przejść przez różne sieci na swojej drodze. Zewnętrzna „koperta" zawsze zawiera informacje niezbędne do podróży do następnej sieci. Po dotarciu do niej router zdejmuje jedną warstwę pakietu (zewnętrzną kopertę) i nakłada kolejną, zawierającą informację na temat podróży do następnej sieci. Kolejne warstwy wewnętrzne zawierają coraz bardziej szczegółowe informacje na temat adresu komputera w Internecie oraz procesu na komputerze będącym adresatem pakietu. Informacje te obejmują również różne typy wiadomości na temat priorytetu przenoszonych danych oraz inne dane pomagające w bezpiecznej przesyłce.

Narzut transmisyjny: komunikacja połączeniowa i bezpołączeniowa

W komunikacji sieciowej istnieje pojęcie nawiązywania połączenia, które polega na zastosowaniu procedur służących do zapewnienia uporządkowanej komunikacji wysokiej niezawodności między systemami. Protokoły o niskiej niezawodności przesyłają dane do adresata bez potwierdzenia powodzenia transmisji. Tego typu komunikacja nosi nazwę bezpołączeniowej. Komunikację wysokiej niezawodności nazywamy połączeniową. Protokołem bezpołączeniowym z rodziny TCP/IP jest protokół *UDP* (*User Datagram Protocol*), natomiast protokół połączeniowy nosi nazwę *TCP* (*Transmission Control Protocol*).

Adresy IP

Każdy interfejs sieciowy w Internecie posiada własny adres IP. Jeśli założymy, że wykorzystywany serwer posiada tylko jedną kartę sieciową, będzie on właścicielem co najmniej jednego adresu IP. W przypadku zastosowań domowych i małych form adres IP jest nadawany przez dostawcę usług internetowych. W firmach adres ten nadaje z reguły administrator sieci.



Kiedy stosujemy mechanizm translacji adresów (*NAT*), wykorzystywane są adresy prywatne. Jeśli używasz tego typu konfiguracji, najprawdopodobniej wiesz wystarczająco dużo, by spokojnie pominąć dalszy ciąg wywodów na temat adresów, lub wpędzasz się w kłopoty, z których będziesz mógł wybrnąć tylko przy pomocy dobrze opłaconego specjalisty.

Adres IP jest zapisany w postaci czterech liczb o wartościach od 0 do 255. Typowy adres IP ma następującą postać: 192.168.1.14. Nie wszystkie kombinacje liczb dają użyteczne adresy IP, lecz nie musimy się tym przejmować, ponieważ to nie my je wymyślamy.

Każdy pakiet skierowany do naszego systemu będzie miał wyspecyfikowany jego adres IP w polu swojego adresu docelowego. Każdy pakiet wychodzący z systemu będzie miał jego adres w swoim polu adresu źródłowego. Te same zasady dotyczą innych systemów w sieci, których ruch będziemy monitorować z naszego serwera. System NIDS można skonfigurować w taki sposób, że będzie monitorować określone adresy źródłowe i docelowe (w dowolnych kombinacjach).

Adresy dynamiczne

Adres IP skonfigurowany na stałe w systemie nazywamy adresem statycznym. Wiele organizacji wykorzystuje odmienne podejście polegające na dynamicznym nadawaniu adresów IP z predefiniowanej puli podczas uruchamiania systemu. Ten mechanizm obsługuje protokół *DHCP* (*Dynamic Host Configuration Protocol*). Wykorzystanie tego typu mechanizmu utrudnia konfigurację systemu NIDS, ponieważ nie można określić adresów IP systemów, które miałyby być śledzone przez ten system (mogą być różne po każdym uruchomieniu systemu).

DNS

Niewiele osób ma wprawę w zapamiętywaniu czwórek liczb od 0 do 255 (choć podczas pracy z systemami NIDS można w tym dojść do wprawy). Ludzi preferują nazwy, takie jak „Alicja", „Dawid" czy „www.yahoo.com". System *DNS* (*Domain Name System*) jest sposobem na związanie adresów IP z nazwami oraz na udostępnienie wszystkim zainteresowanym nazw systemów podłączonych do Internetu. W ten sposób każda organizacja może zarządzać listą nazw własnych serwerów podłączonych do Internetu.

Gdy system zna nazwę systemu docelowego, najczęściej komunikuje się z serwerem DNS w celu uzyskania odpowiedniego adresu IP, na który może przesłać dane. Zwróćmy uwagę, że żądanie DNS jest samo w sobie również transakcją, odbywającą się przed

nawiązaniem transakcji przesyłania danych. Jeśli wykorzystywany program wypisuje informację na temat adresu IP wraz z nazwą (wiele systemów NIDS działa właśnie w ten sposób), częste żądania DNS mogą spowolnić wyświetlanie informacji. Zalecamy wyłączenie zapytań DNS tam, gdzie to tylko możliwe zarówno w aplikacjach, jak i w poleceniach systemowych.

Routery

Komunikacja (pakiety) na swojej drodze od nadawcy do adresata przechodzi przez różne urządzenia zwane *routerami* i *bramami*. Router po prostu przesyła pakiet z jednej sieci do drugiej. W organizacji średniej wielkości komunikacja wewnętrzna najczęściej jest przekazywana za pośrednictwem jednego do czterech takich urządzeń. Zewnętrzna komunikacja przechodzi przez 10 do 20 routerów.

W naszych rozważaniach używamy określeń „sieć lokalna" oraz „inne sieci". Jednak zaawansowana technologia sieciowa nieco rozmywa granice między sieciami, lecz nadal w miarę konsekwentnie można twierdzić, że granicę między różnymi sieciami stanowi router.

Router brzegowy jest ostatnią barierą między siecią lokalną a Internetem. Routery posiadają dwa lub większą liczbę interfejsów sieciowych, każdy z nich przyłączony jest do innej sieci. Jeśli pakiet wysłany z sieci lokalnej może dotrzeć do adresata bez potrzeby przechodzenia przez router brzegowy, oznacza to, że adresat również znajduje się w sieci lokalnej. Kilka lat temu z dowolnej stacji można było monitorować każdy pakiet przychodzący do sieci lokalnej, nawet jeśli był przeznaczony do innego adresata. W wielu sieciach wykorzystujących przestarzały lub niedrogi sprzęt nadal jest to możliwe.

Coraz popularniejsze stają się jednak *sieci przełączane*. W sieciach tego typu dane, które docierają do danego interfejsu sieciowego, są tylko do niego adresowane. Ruch nieadresowany do danego interfejsu nigdy do niego nie dotrze. Jeśli chcemy monitorować ruch w sieci, administrator sieci może skonfigurować specjalny port na przełączniku (tzw. *monitor port*), na który będzie dodatkowo przesyłany cały ruch adresowany do pozostałych stacji przyłączonych do przełącznika. Jeśli na przykład wykryjemy podejrzany ruch w podsieci, który wydaje się pochodzić z zewnątrz, lecz ruch ten nie jest wykrywany na routerze brzegowym, mamy jasny dowód na to, że któryś z systemów w sieci został przejęty przez włamywacza, który fałszuje adres źródłowy wysyłanych przez siebie pakietów.

Żyj jak najlepiej z innymi administratorami!

Jak można zauważyć, wiele rzeczy, które istnieją w sieci i są nam potrzebne, zależy od administratora tej sieci. W rzeczywistości dobre poznanie swojego administratora i jego pracy jest jedną z lepszych inwestycji. Jeśli poważnie traktujesz wykorzystanie mechanizmów NIDS, bardzo ważne jest nadążanie za zmianami w sieci, często niezauważalnymi dla zwykłych użytkowników. Informacje te posiada administrator sieci, ponieważ to on wdraża zmiany!

Gdy coś pójdzie źle, nie obrażaj administratora, wytykając mu niekompetencję. Aby wykonać swoją pracę, musicie koegzystować. Pamiętaj, że administrator sieci prawdopodobnie myśli o Tobie to samo, co Ty myślisz o swoich użytkownikach.

Podsieci (notacja CIDR)

Przy dokładniejszym spojrzeniu pojęcie podsieci nieco się rozmywa. Podsieć to nadal sieć. Termin podsieć oznacza po prostu, że jest rozumiana jako część większej sieci. Sieć jest z reguły podzielona na podsieci za pomocą routerów lub przełączników, zatem wcześniejsze informacje na temat routerów odnoszą się również do podsieci.

W podsieci musimy znać:

- jej zakres adresów IP,
- adres IP routera, który muszą wykorzystywać systemy w sieci w celu komunikacji z innymi sieciami. Jest to szczególnie ważna informacja, gdy jednym z tych systemów jest serwer!

Zakresy adresów IP w podsieci najczęściej określane są za pomocą notacji CIDR. Skrót *CIDR* pochodzi od *Classless Inter-Domain Routing*. Jeśli w notacji CIDR określimy podsieć w następujący sposób: `192.168.249.42/23`, oznacza to, że adresy IP w tej podsieci muszą mieć pierwsze 23 bity takie same jak w adresie `192.168.249.42`. Adres podsieci `192.168.249.42` ma następującą reprezentację dwójkową:

```
11000000 10101000 11111001 00101010
```

Zatem każdy adres IP z pierwszymi 23 bitami takimi, jak w powyższej masce należy do podsieci. Zatem zakres adresów obejmuje adresy od następującego:

```
11000000 10101000 11111000 00000000
```

do następującego:

```
11000000 10101000 11111001 11111111
```

Ten sam zakres w notacji IP obejmuje adresy od `192.168.248.0` do `192.168.249.255`.

Podobnie podsieć `192.168.249.42/29` obejmuje adresy od `192.168.249.40` do `192.168.249.47`.

Porty

Unikalne adresy IP są podstawą komunikacji między systemami w Internecie. Jednak komunikacja tak naprawdę nie odbywa się między komputerami, a między procesami działającymi w tym przypadku na różnych komputerach. Dostarczenie pakietu do komputera to nie wszystko, pakiet musi zostać dostarczony do odpowiedniego procesu, który będzie wiedział, co z nim zrobić.

Każdemu procesowi komunikującemu się w sieci nadawany jest unikalny numer portu z przedziału od 1 do 65535. Każdy pakiet przesyłany na dany adres, w którym wartość w polu *Destination Port* (port docelowy) pasuje do portu danego procesu, zostanie dostarczony właśnie do niego. Oczywiste jest, że każdy pakiet przesłany przez ten proces będzie miał w polu *Source Port* (port źródłowy) określony ten sam numer portu.

Wcześniej wspomnieliśmy o różnicach między protokołami TCP i UDP. Protokoły te różnią się od siebie na tyle, że ten sam numer portu może być wykorzystany przez dwa

różne procesy, jeśli tylko jeden z nich wykorzystuje protokół TCP, a drugi — UDP. Egzemplarze tego samego portu dla protokołu TCP i dla protokołu UDP są uważane za dwa różne porty. Pierwszy z nich określamy jako `<numer>/tcp`, a drugi jako `<numer>/udp`.

Zgodnie z przyjętą konwencją porty od 1 do 1024 są na stałe przypisane określonym usługom sieciowym, na przykład proces serwera SMTP będzie wykorzystywać port `25/tcp`, serwer WWW udostępnia swoje usługi na porcie `80/tcp`, usługi DNS są dostępne na portach `53/tcp` oraz `53/udp` (wybór protokołu uzależniony jest od tego, jakie informacje są odczytywane z serwera DNS) itd.

Procesy, gniazda i wiązania

W poprzednim podrozdziale napisaliśmy, że serwer SMTP wykorzystuje port `25/tcp`, co oznacza, że proces serwera (na przykład sendmail) musi nasłuchiwać na tym porcie w oczekiwaniu na połączenia. W tym celu podczas uruchomienia proces serwera wywołuje specjalną funkcję systemową `bind()`, która spowoduje, że port `25/tcp` będzie związany z procesem serwera nawet wtedy, kiedy nie jest na nim aktywne żadne połączenie. Proces następnie przechodzi do stanu nasłuchiwania (`LISTENING`), w którym jest gotowy do przyjmowania połączeń. Proces pozostaje w stanie nasłuchiwania aż do momentu, gdy otrzyma połączenie lub zostanie zakończony, w wyniku czego zwolni port.

System Linux, podobnie jak inne systemy klasy UNIX, traktuje wszystkie obiekty, tak jak pliki. W celu przypisania sobie portu proces prosi system Linux o ustanowienie tzw. gniazda. Gniazdo jest po prostu jedną ze stron komunikacji sieciowej. Linux zwraca tzw. deskryptor pliku gniazda, który może być użyty przez proces, podobnie jak byłby użyty zwykły plik, z tą różnicą, że w przypadku gniazd istnieje większa liczba opcji konfiguracyjnych. Następnie proces wywołuje funkcję `bind()` wiążącą z tym gniazdem określony port.

Proces nasłuchiwania można monitorować z zewnątrz. Gdy system nasłuchuje na określonym porcie, na przykład `25/tcp`, mówi się, że port ten jest otwarty. Ponieważ port ten jest przez konwencje dedykowany serwerowi SMTP, zakłada się, że na tym porcie nasłuchuje właśnie serwer tego typu (na przykład sendmail). Gdy na tym porcie pojawi się połączenie, serwer wysyła informację tekstową informującą o swojej wersji i tym podobne dane (czasem dane te są zbyt szczegółowe!).

Z samego komputera można odczytać dużo więcej informacji. Polecenie `netstat -apn` wypisuje stan wszystkich aktywnych gniazd, w tym gniazd w stanie nasłuchiwania (`LISTENING`). Opcja `-a` spowoduje, że w wyniku zostaną uwzględnione porty nasłuchujące, opcja `-p` spowoduje wypisanie identyfikatora (PID) oraz nazwy procesu, z którym jest związany dany port, natomiast opcja `-n` zapobiega wykonywaniu zapytań DNS (*DNS lookup*), o czym wspominaliśmy wyżej.

Oto przykładowy wynik polecenia `netstat -apn`:

```
# netstat -apn
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address     Foreign Address      State       PID/Program name
tcp        0      0 0.0.0.0:32768     0.0.0.0:*            LISTEN      495/rpc.statd
tcp        0      0 0.0.0.0:514       0.0.0.0:*            LISTEN      619/xinetd
tcp        0      0 0.0.0.0:5666      0.0.0.0:*            LISTEN      619/xinetd
```

```
tcp        0      0 0.0.0.0:6019            0.0.0.0:*               LISTEN      2955/sshd
tcp        0      0 0.0.0.0:6020            0.0.0.0:*               LISTEN      31991/sshd
tcp        0      0 0.0.0.0:6021            0.0.0.0:*               LISTEN      13355/sshd
tcp        0      0 0.0.0.0:645             0.0.0.0:*               LISTEN      464/ypbind
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      449/portmap
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN      619/xinetd
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      629/sshd
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN      619/xinetd
tcp        0      0 0.0.0.0:6010            0.0.0.0:*               LISTEN      7840/sshd
tcp        0      0 0.0.0.0:6011            0.0.0.0:*               LISTEN      10507/sshd
tcp        0      0 0.0.0.0:6012            0.0.0.0:*               LISTEN      32054/sshd
tcp        0      0 0.0.0.0:6013            0.0.0.0:*               LISTEN      9281/sshd
tcp        0      0 0.0.0.0:6014            0.0.0.0:*               LISTEN      10437/sshd
tcp        0      0 0.0.0.0:6015            0.0.0.0:*               LISTEN      492/sshd
tcp        0      0 10.100.60.38:6011       10.100.60.38:54804      ESTABLISHED 10507/sshd
tcp        0      0 10.100.60.38:6011       10.100.60.38:54800      ESTABLISHED 10507/sshd
tcp        0      0 10.100.60.38:23         10.100.202.234:2253     ESTABLISHED 29887/in.telnetd
tcp        0     52 10.100.60.38:22         10.100.82.66:1030       ESTABLISHED 6338/sshd
tcp        0      0 10.100.60.38:22         10.100.99.81:50237      ESTABLISHED 10437/sshd
tcp        0      0 10.100.60.38:22         10.100.179.83:1511      ESTABLISHED 3015/sshd
tcp        0      0 10.100.60.38:23         10.100.100.220:2115     ESTABLISHED 4001/in.telnetd
tcp        0      0 10.100.60.38:23         10.100.110.70:35857     ESTABLISHED 17750/in.telnetd
tcp        0      0 10.100.60.38:23         131.118.161.29:41387    ESTABLISHED 9512/in.telnetd
tcp        0      0 10.100.60.38:6020       10.100.60.38:47910      ESTABLISHED 31991/sshd
tcp        0      0 10.100.60.38:23         10.100.110.76:1034      ESTABLISHED 14735/in.telnetd
tcp        0      0 10.100.60.38:22         68.50.77.205:1082       ESTABLISHED 28390/sshd
```

Zwróćmy uwagę, iż procesy nasłuchujące mają ustawiony adres IP na `0.0.0.0`. Niektóre systemy mogą mieć kilka adresów IP. Każdy system ma przynajmniej dwa, jednym z nich jest adres tzw. pętli zwrotnej (*loopback*): `127.0.0.1`. Adres `0.0.0.0` jest interpretowany jak wszystkie adresy w systemie. Oznacza to na przykład, że proces `sshd` nasłuchujący na porcie `22/tcp` będzie akceptował połączenia zarówno przychodzące na adres `10.100.60.38`, jak i `127.0.0.1` (oraz inne adresy, które mogłyby być przypisane systemowi).

lsof

Kolejny program, *lsof*, wypisuje nawet większą ilość informacji. Wywołanie programu *lsof* z nazwą pliku spowoduje wypisanie nazwy procesu aktualnie wykorzystującego dany plik. Opcja `-i` ogranicza wypisywane procesy do wykorzystujących gniazda danego typu. W celu uzyskania tych informacji należy wywołać polecenie `lsof` jako *root*:

```
# lsof -i tcp
COMMAND    PID USER FD   TYPE   DEVICE SIZE NODE NAME
sshd      7313 root  4u IPv4 64526277       TCP me.my.org:ssh->ppp-199.dial up.com:1975
➥(ESTABLISHED)
xinetd    7334 root  3u IPv4     2590       TCP *:ftp (LISTEN)
xinetd    7334 root  4u IPv4     2591       TCP *:nrpe (LISTEN)
xinetd    7334 root  7u IPv4     2593       TCP *:shell (LISTEN)
xinetd    7334 root  9u IPv4     2595       TCP *:telnet (LISTEN)
xinetd    7334 root 10u IPv4 64526304       TCP me .my.org:ftp->ppp-191.dialup.com:2501
➥(ESTABLISHED)
xinetd    7334 root 11u IPv4 64526305       TCP m.my.org:54740->ftp.my.org:ftp
➥(ESTABLISHED)
```

```
in.telnet 7475 root  0u IPv4 63969514       TCP m.my.org:telnet->pc.cb.net:1964
➥(ESTABLISHED)
in.telnet 7475 root  1u IPv4 63969514       TCP m.my.org:telnet->pc.cb.net:1964
➥(ESTABLISHED)
in.telnet 7475 root  2u IPv4 63969514       TCP m.my.org:telnet->pc.cb.net:1964
➥(ESTABLISHED)
sshd      7840 root  4u IPv4 63970291       TCP m.my.org:ssh->pc.nasa.gov:37606
➥(ESTABLISHED)
```

W wyniku swojego działania program *lsof* wypisał jako plik każde z gniazd, nazwę procesu wykorzystującego gniazdo, identyfikator tego procesu oraz jego właściciela. Informacje te mogą być bardzo użyteczne do określenia procesu inicjującego połączenie sieciowe, którego nie powinno być w systemie i którego pochodzenie jest trudne do określenia.

ICMP

Protokół *ICMP* (*Internet Control Message Protocol*) wykorzystuje pakiety w charakterze stosunkowo niewielkich komunikatów służących do celów informacyjnych oraz do zarządzania siecią. Wiele z tych komunikatów pozwala rozeznać sytuację problemu z funkcjonowaniem systemu lub usługi oraz jego przyczyny. Na przykład podczas wysłania pakietu do nieistniejącego systemu, router danej podsieci odeśle pakiet ICMP typu `3` (*Destination Unreachable*) o kodzie `1` (*Host Unreachable*).

Najlepiej znanym zastosowaniem protokołu ICMP jest polecenie `ping`, wysyłające pakiety ICMP typu `8` (*Echo Request*), po których oczekuje w odpowiedzi pakietów typu `0` (*Echo Reply*).

UDP

Jak pamiętamy, *UDP* (*User Datagram Protocol*) jest protokołem komunikacji bezpołączeniowej z rodziny protokołów TCP/IP. Protokół ten jest wykorzystywany zwykle wtedy, gdy na drodze komunikacji nie oczekujemy wielu routerów (lub żadnego). Innymi słowy, UDP jest wykorzystywane do realizacji usług, takich jak NFS, które z reguły są wykorzystywane w sieciach lokalnych.

Z powodu braku połączenia w komunikacji UDP nagłówek pakietu jest stosunkowo niewielki. Dla uproszczenia przyjmijmy, że nagłówek ten zawiera jedynie adresy oraz porty źródłowe i docelowe pakietu.

TCP

TCP jest protokołem wykorzystującym połączenia. Nagłówek pakietu zawiera sporo informacji wykorzystywanych w celu kontroli przekazania pakietu oraz w celu odtworzenia pełnych danych z poszczególnych pakietów składowych. Protokół TCP zapewnia odtworzenie pakietów w odpowiedniej kolejności nawet wtedy, gdy do odbiorcy docierały one w kolejności zaburzonej (dwa kolejne pakiety mogą teoretycznie przemieszczać się różnymi trasami, więc docieranie ich do adresata w nieoryginalnej kolejności

nie jest niczym nadzwyczajnym). Protokół TCP zapewnia również integralność przesyłanych danych. Ponadto ma on wbudowany mechanizm wysyłania żądań ponownego przesłania na przykład uszkodzonych pakietów. Te procedury zapewniające poprawność transmisji są cechą charakterystyczną protokołów połączeniowych. Połączenia TCP w rzeczywistości rozpoczynają się wymianą pakietów złożonych z samych nagłówków bez żadnej zawartości w celu nawiązania komunikacji i ustalenia parametrów transmisji.

Połączenia TCP są wykorzystywane do komunikacji na „długich dystansach" (w ujęciu sieciowym), na przykład przy usługach FTP oraz SSH.

TCP jest protokołem *zachowującym stan*. W UDP dane są wysyłane i tutaj kończy się troska nadawcy. Protokół ten nie śledzi stanu transmisji, ponieważ przy transmisji bezpołączeniowej nie ma kontroli dostarczenia pakietu. W protokole TCP sprawdzany jest stan przekazania każdego pakietu. Z tego powodu TCP musi „pamiętać" wysłane pakiety, aby potwierdzić sobie stan poprawnej wysyłki po otrzymaniu potwierdzenia (lub wysłać pakiet ponownie w razie problemów).

Taka pamięć nazywana jest stanem. Po wysyłce pakietu TCP znajduje się w stanie oczekiwania na potwierdzenie (*awaiting receipt*). Po potwierdzeniu dotarcia pakietu TCP przechodzi w stan „nieoczekujący potwierdzenia" (*not awaiting receipt*). Oprócz tych dwóch podstawowych stanów TCP wykorzystuje inne, które pozwalają na dokładną kontrolę poprawności dostarczania danych.

SYN, ACK, FIN

W nagłówkach TCP występują tzw. bity znaczników (*flag bits*) — bity włączone lub wyłączone, informujące o stanie komunikacji na obydwu jej końcach. Bity znaczników są określane na podstawie ich pozycji w nagłówku. W niniejszym podrozdziale zapoznamy się ze znacznikami SYN, ACK oraz FIN.

Znacznik SYN oznacza, że dany pakiet ma na celu zainicjowanie sesji komunikacyjnej. Znacznik ten znacza zatem mniej więcej tyle, co „chciałbym z tobą rozmawiać". Znacznik ACK z kolei, że dany pakiet zawiera potwierdzenie otrzymania poprzedniego pakietu. Oznacza on tyle, co „usłyszałem cię".

W celu zainicjowania sesji system A wysyła do systemu B pakiet składający się z samych nagłówków. W nagłówku będzie ustawiony tylko znacznik SYN. W odpowiedzi system B wysyła do systemu A pakiet spełniający dwie funkcje: po pierwsze, informuje za pomocą ustawionego znacznika ACK o tym, że pakiet inicjujący doszedł. Po drugie, informuje system A, iż system B jest gotowy do nawiązania połączenia, w tym celu ustawiony jest w nim również znacznik SYN.

Po otrzymaniu tego pakietu system A wysyła do B pakiet potwierdzający z ustawionym tylko znacznikiem ACK. Wtedy nie jest już ustawiony znacznik SYN. W tym momencie obydwa systemy już zakomunikowały, że chcą rozmawiać, więc nie ma potrzeby dalszego informowania o tym fakcie.



Ta wymiana pakietów SYN, SYN-ACK oraz ACK nazywana jest potrójną inicjacją (*triple handshake*) połączenia TCP.

Gdy system A nie ma już nic do wysłania, przekazuje systemowi B pakiet z ustawionym znacznikiem FIN. W pakiecie tym jest ustawiony również znacznik ACK, potwierdzający otrzymanie poprzedniego pakietu. System B potwierdza zakończenie połączenia, wysyłając pakiet ze znacznikiem ACK. Protokół TCP uwzględnia również sytuację, w której system A chce zakończyć komunikację, lecz system B nadal ma dane do wysyłki. W tej sytuacji połączenie, nazywane półzamkniętym (*half-closed*), pozostanie w tym stanie, do momentu gdy system B wyśle do systemu A pakiet z informacją FIN-ACK, a system A odpowie mu na to pakietem ze znacznikiem ACK.

Powyższy opis procedur komunikacyjnych miał na celu przybliżenie niektórych szczegółów komunikacji TCP, a także posłużył jako podstawa do konfiguracji systemu NIDS.

Numerowanie znaczników SEQ i ACK

Protokół TCP obsługuje numerację sekwencji `SEQ` oraz potwierdzeń `ACK`. Sytuacja ta może się nieco skomplikować. W uproszczeniu system A upewnia się, iż system B otrzymał pakiety w odpowiedniej kolejności, nadając im kolejne numery. Gdy system B otrzymuje pakiet, odsyła systemowi A pakiet z potwierdzeniem (znacznik ACK) zawierający liczbę potwierdzającą równą numerowi pakietu plus jeden. Oznacza to, że w następnym pakiecie wysyłanym przez system B zostanie wysłana liczba potwierdzająca ostatnio otrzymany pakiet przesłany przez A.

System B to samo robi z wysyłanymi przez siebie pakietami. Każda ze stron komunikacji przy inicjacji wybiera niezależnie początkowy numer sekwencji, zatem system B rozpoczyna komunikację z innym numerem sekwencji od systemu A. Dobrze zaimplementowane protokoły komunikacyjne wykorzystują jak najbardziej przypadkowe początkowe numery sekwencji, co zabezpiecza przed próbami przejęcia sesji TCP.

Rozpoznania i zagrożenia

Poznaliśmy podstawy funkcjonowania rodziny protokołów TCP/IP, nadszedł czas, aby omówić niektóre sposoby wykorzystania ich przez napastników. Ogólnie mówiąc, działania podejmowane przez napastników można z grubsza podzielić na *rozpoznania* i *zagrożenia*.

Rozpoznanie (lub skanowanie) oznacza próbę zdobycia większej ilości informacji na temat systemu bez wyrządzania mu krzywdy. W rzeczywistości większość rozpoznań ma na celu zwrócenie na siebie jak najmniejszej uwagi, aby uniknąć wykrycia. Zagrożeniem jest każda próba wyrządzenia szkody lub przejęcia kontroli nad systemem.

Skanowanie portów SYN oraz SYNFIN

Skanowanie typu SYN jest bardzo powszechne. Wykorzystuje one protokół potrójnej inicjacji. Jak pamiętamy, system A wysyła do systemu B pakiet z ustawionym znacznikiem `SYN`, oczekując pakietu z ustawionymi znacznikami `SYN` i `ACK`. Załóżmy, że system A nie dokończy sekwencji inicjacji. System B odczeka określony czas i zamknie

połączenie ,ponownie przechodząc w stan oczekiwania na następny pakiet SYN. System A jednak już wie, że port w systemie B, z którego otrzymał pakiet SYN-ACK, jest otwarty.

Gdy system A wyśle kolejno takie pakiety SYN do wszystkich portów poniżej 1024, zgromadzi informacje na temat usług udostępnianych przez system B. Dane na temat tych usług można wywnioskować z odpowiedzi systemu B. Zamiast jednak potwierdzić połączenie, system A wysyła pakiet z ustawionym znacznikiem RST (*reset*), który natychmiast kończy połączenie. W trakcie tego procesu system A dodatkowo otrzymuje próbkę inicjujących numerów sekwencji, na podstawie których może określić poziom ich przypadkowości. Ponieważ sekwencja potrójnej inicjacji nie dobiega końca, żadna z usług w systemie B nie zapisze w dziennikach informacji o połączeniu z systemu A.

Skanowanie typu SYNFIN wykorzystuje głębiej ukrytą przypadłość protokołu TCP. Protokół ten nie przewiduje bowiem możliwości wystąpienia „niemożliwych" kombinacji znaczników. Pakiet zawierający znacznik SYN („chcę rozpocząć sesję komunikacyjną") oraz FIN („chcę zakończyć tę sesję komunikacyjną") nie ma określonego znaczenia w definicji protokołu. Z powodu, że nie istnieje określone zachowanie w odpowiedzi na pakiet tego typu, sposób, w jaki zachowa się system B, zależy od implementacji w nim mechanizmów sieciowych. Jedno jest pewne w większości, a może nawet we wszystkich systemach z rodziny UNIX — odpowiedź na taki pakiet zależy od tego, czy dany port jest otwarty, czy nie. Sposób reakcji na pakiet SYNFIN daje sporo informacji na temat systemu operacyjnego, czasem nawet na temat numeru wersji lub konkretnej wersji łatki. Ponadto napastnik uzyskuje informacje na temat otwartych portów.

Ataki typu odmowa usługi (DoS)

Ataki typu odmowa usługi (*DoS — Denial of Service*) nie stanowią prób włamania do systemu, lecz raczej próbę zablokowania usługi udostępnianej przez atakowany system.

Czasem zdarza się, że atak typu DoS powoduje zawieszenie całego systemu. Najprostsze ataki typu DoS to ataki „zalewające" (*flood*), polegające na wysyłaniu do usługi większej ilości ruchu, niż jest ona (lub cały system) w stanie obsłużyć. Jednym z przykładów takiego ataku jest „bombardowanie pocztowe" (*mail bombing*) serwera pocztowego lub skrzynki konkretnego użytkownika.

Nieco bardziej wyrafinowaną formą ataku typu DoS jest zalewanie pakietami SYN (*syn flooding*). Wcześniej opisaliśmy mechanizm wykorzystania pakietów SYN w celu wykrycia otwartych portów bez nawiązywania połączenia (system ofiary odpowie sekwencją SYN-ACK, a następnie zaczeka na pakiet ACK, gdy go nie otrzyma w określonym czasie, zamknie połączenie). Załóżmy teraz, że system A wysyła tysiące pakietów SYN do systemu B i kontynuuje ich wysyłkę bez przerwy. W pewnym momencie wyczerpią się zasoby sieciowe systemu B, ponieważ będzie on oczekiwać na tysiące pakietów ACK, które nie nadejdą. System B zostanie w ten sposób odcięty od Internetu, ponieważ przez ten zalew nie przedrą się żadne prawdziwe połączenia.

Inne rodzaje ataków typu DoS opisujemy w podrozdziale „Ataki przepełnienia bufora".

Ataki wykorzystujące błędy w systemie

Programy piszą ludzie, nie maszyny. Linux też jest napisany przez ludzi. Mają oni nieznośną wadę braku konsekwencji i popełniania błędów w projekcie lub implementacji — jest to jedna z cech twórczych umysłów. Gdy taki błąd zostanie wykorzystany przez kogoś innego w celu zmuszenia systemu do wykonania operacji nieprzewidzianych przez jego twórcę, mamy do czynienia z dziurą bezpieczeństwa.

Ta historia przytrafiła się naprawdę. Jedna z wcześniejszych wersji serwera sendmail posiadała polecenie `wizard` (czarownik). Po nawiązaniu połączenia na porcie `25/tcp` i przesłaniu do serwera polecenia `WIZ` system odpowiadał komunikatem „Please pass, oh mighty wizard" (czyli: Proszę wejdź, o potężny czarowniku) i użytkownik uzyskiwał dostęp z prawami konta *root* do całego systemu. Polecenie `WIZ` było zapewne ciekawą funkcją wspomagającą proces diagnostyki systemu, lecz z pewnością wielkim przeoczeniem było pozostawienie jej w oficjalnej wersji programu! Zdobycie wiedzy na temat polecenia `WIZ` nie było trudne, szczególnie w sytuacji gdy kod źródłowy systemu był powszechnie dostępny. Jako jedyny komentarz pozostawmy to, że obecne wersje systemu sendmail nie posiadają już tej wspaniałej funkcji.

Ataki przepełnienia bufora

Gdy w programie zostaje wpisana niewielka ilość informacji, dane te są zwykle przechowywane w pamięci operacyjnej, na przykład podczas logowania się w systemie program *login* do momentu podania hasła zapisuje w pamięci nazwę użytkownika. Zatem program wykorzystuje dodatkową pamięć oprócz tej, która służy w celu przechowywania kodu wykonywalnego i danych początkowych. Nic w tym nadzwyczajnego.

Ile pamięci potrzebuje program w celu przechowywania nowych danych? Identyfikator użytkownika średnio ma rozmiar ośmiu bajtów, hasło prawie na pewno poniżej 15 bajtów. Oszczędnie napisany program logowania do systemu przewidzi zatem około 30 bajtów na nazwę użytkownika i hasło. Co się jednak stanie, gdy ktoś wprowadzi 31-znakowy napis jako nazwę użytkownika? Porządnie napisany kod zliczy bajty przed zapisem ich w pamięci i odmówi przyjęcia ostatniego bajtu aż do momentu opróżnienia bufora. Zdumiewająca ilość kodu nie jest jednak porządnie napisana.

Gdy do takiego programu wprowadzane są dane o rozmiarze większym od rozmiaru zaalokowanej pamięci, a program nadal je przyjmuje, dane te zaczynają nadpisywać inne obszary pamięci. Takie wykroczenie programu poza obszar zaalokowanej pamięci nazywamy przepełnieniem bufora. Gdy taki program działa z prawami konta *root*, może bez przeszkód nadpisać każdy obszar pamięci w systemie. Przepełnienie w nim bufora może nawet nadpisać kod wykonywalny systemu operacyjnego, co prowadzi do niekontrolowanego restartu lub zawieszenia systemu. Zaawansowane wykorzystanie przepełnienia bufora polega na umieszczeniu w danych wejściowych odpowiednio spreparowanych danych w taki sposób, że zastąpią one odpowiednią część systemu operacyjnego i pozwolą w sposób kontrolowany przejąć przez napastnika prawa błędnie napisanego programu w systemie (czyli często prawa konta *root*). Pierwszy opisany typ przepełnienia bufora jest atakiem typu DoS, drugi zaś — udanym wykorzystaniem błędu w systemie do włamania.

Ataki wykorzystujące aplikacje

Do tej pory nie braliśmy pod uwagę danych przekazywanych w pakietach. W końcu to jest główna przyczyna istnienia pakietów. Dane mogą być komunikatami przekazywanymi między serwerami, adresem URL wysyłanym do serwera WWW lub praktycznie dowolnymi danymi przeznaczonymi dla aplikacji sieciowej.

Demony *httpd* i *sshd*, podobnie jak oprogramowanie komunikacji sieciowej, są podatne na ataki wykorzystujące błędy w programowaniu.

Błędnie skonfigurowany serwer WWW pozwoli na uzyskanie dostępu do katalogów systemowych za pomocą adresu URL zawierającego elementy ścieżki typu *../../...* W ten sposób można wskazać pliki i katalogi spoza drzewa katalogów serwera WWW, na przykład *../../etc/passwd*. Niedawno odkryto niewielki błąd przepełnienia bufora w serwerze *sshd*. Niemal natychmiast pojawił się sposób uzyskania praw konta *root* za pomocą tego błędu. Otrząśnięcie się po tym potknięciu trwało sporo czasu.

Ataków tego typu nie można wykryć, analizując nagłówki pakietów, ponieważ ataki zawarte są w samych danych przesyłanych przez te pakiety.

Projektowanie defensywnej konfiguracji sieci

Przed rozpoczęciem konfiguracji systemu NIDS należy poznać konfigurację sieci. W tym podrozdziale zapoznamy się z wybranymi strategiami wykorzystywanymi do budowania stosunkowo bezpiecznych sieci.

Filtrowanie na routerach oraz zapory sieciowe

Jak wiemy, routery są „bramami" między sieciami. Wiele routerów ma możliwość blokowania określonych typów pakietów wysyłanych z sieci lub do niej. Jeśli wiemy, że określony system powoduje problemy, możemy skonfigurować router w taki sposób, że będzie odrzucał wszelkie pakiety z adresem źródłowym wskazującym na ten komputer. Gdy jeden z komputerów wymaga szczególnej ochrony, router może filtrować pakiety z adresem źródłowym „złośliwego" hosta i adresem docelowym systemu wymagającego szczególnej ochrony. Gdy chcemy chronić określone usługi, możemy filtrować tylko wybrane porty docelowe itd.

Zapora sieciowa (*firewall*) jest specjalnym systemem pozwalającym na definiowanie reguł przepuszczania i blokowania pakietów przy użyciu bardziej zaawansowanych zestawów reguł, niż pozwala na to router. Zaawansowana zapora sieciowa potrafi blokować skany typu SYN i zapobiega zalewowi pakietów dzięki kontroli stanu połączeń. Zapora sieciowa może na przykład w imieniu systemu w sieci lokalnej negocjować połączenie z systemem poza siecią, następnie negocjować połączenie z systemem w sieci, a w końcu zestawić te dwa połączenia pośrednie w jedno wirtualne połączenie.

Oczywiście zapora sieciowa nadal będzie podatna na atak typu zalewu pakietami SYN, lecz w tym przypadku dodatkowe zasoby muszą być zabezpieczone tylko na zaporze, a nie na wszystkich systemach z dostępem do Internetu.

Filtrowanie ingress i egress

Jest to rodzaj prostych reguł, których stosowanie w zaporze sieciowej lub routerze jest bardzo zalecane. Załóżmy, że posiadamy sieć lokalną o adresach `10.20.400.xxx` lub w notacji CIDR — `10.20.400.0/24`

- **Reguła 1**. Na interfejsie zewnętrznym blokujemy wszystkie przychodzące pakiety z adresami źródłowymi z sieci `10.20.400.xxx`. To jest filtr *ingress*.
- **Reguła 2**. Na interfejsie wewnętrznym blokujemy wszystkie przychodzące pakiety z adresami źródłowymi spoza sieci `10.20.400.xxx`. To jest filtr *egress*.

Filtrowanie *ingress* zabezpiecza przed pakietami udającymi pakiety pochodzące z sieci lokalnej, lecz naprawdę przychodzące z zewnątrz. Filtrowanie *egress* zabezpiecza systemy spoza naszej sieci przed pakietami wysyłanymi przez komputery z sieci lokalnej, lecz udającymi systemy z innej sieci.

Zalety filtrowania *ingress* są oczywiste. Zalety filtrowania *egress* ujawniają się dopiero wtedy, gdy faceci w czerni wyśledzą źródło ataku na serwer rządowy umiejscowione w naszej sieci, po czym otrzymamy o trzeciej nad ranem telefon od Wielkiego Brata ze stanowczą prośbą wytłumaczenia tej sytuacji. Zaufaj nam. Skonfiguruj filtrowanie *egress*. Zawczasu.

DMZ

Termin DMZ jest stosunkowo mało obrazowy. Dokładnie oznacza on strefę zdemilitaryzowaną. W naszym kontekście oznacza wydzieloną część sieci między routerem a zaporą sieciową lub między dwoma zaporami. Koncepcja DMZ-tu polega na tym, że wydzielona jest część sieci, która musi być eksponowana w Internecie. Pozostała część sieci jest kryta za dodatkową zaporą i nie jest dostępna bezpośrednio przez systemy z Internetu. Cała komunikacja systemów z sieci prywatnej następuje za pośrednictwem jednego, dobrze zabezpieczonego systemu umieszczonego w DMZ.

Hosty bastionowe

Koncepcja hostów bastionowych wiąże się z koncepcją DMZ-tu. Gdy chcemy udostępnić bezpieczny serwer WWW, jedną ze strategii jest skonfigurowanie serwera udostępniającego wyłącznie usługę WWW. W serwer ten inwestujemy sporo czasu i wysiłku, konfigurując go w bezpieczny sposób za pomocą łat i technik wzmacniających (*hardening*). Ten serwer staje się hostem bastionowym WWW i najprawdopodobniej wyląduje w DMZ. Hosty bastionowe są niezłą strategią nawet wtedy, gdy nie wykorzystuje się konfiguracji typu DMZ.

Dedykowane serwery

Wykorzystanie dedykowanych serwerów stanowi strategię administracyjną i strategię bezpieczeństwa. Koncepcja polega na świadczeniu pojedynczych usług na każdym z serwerów tak, aby serwer FTP udostępniał jedynie usługi FTP, serwer SSH wyłącznie pozwalał na zalogowanie się za pomocą SSH, serwer pocztowy uruchamiał jedynie serwer SMTP itd.

Z punktu widzenia administracji w przypadku konieczności założenia poprawki na serwer FTP nie ma problemu z określeniem komputerów, na których trzeba to zrobić. Nie ma również obaw o efekty uboczne działania takiej poprawki na przykład na nasz serwer HTTP.

Z punktu widzenia bezpieczeństwa mamy lepszy wgląd w sytuację i wiemy, które serwery mogą być podatne na błędy typowe dla serwerów FTP. Co więcej, wiemy, które z serwerów z całą pewnością nie są podatne na tego typu błędy, ponieważ na nich nie są w ogóle zainstalowane żadne programy wykorzystujące protokół FTP. Można również skonfigurować router lub zaporę sieciową w taki sposób, że nie będzie przepuszczać ruchu dotyczącego nieobsługiwanych usług do danych adresów docelowych. Innymi słowy, dla IP należącego do serwera poczty nie ma sensu przepuszczanie pakietów kierowanych na port 21 (FTP).

Oczywiście blokowanie portów, które i tak nie są otwarte, wydaje się nieco przesadne i brzmi jak strategia jednoczesnego używania paska i szelek. W tym dziwnym świecie trudno określić, co to jest nadmiar zabezpieczeń. Z drugiej strony, czy lepiej wykazać za dużo ostrożności, czy za mało?

Po określeniu strategii obronnych nadszedł czas na instalację „alarmu antywłamaniowego" na wypadek, gdyby mimo całych zabiegów zabezpieczających komuś udało się przedostać przez bariery.

Projektowanie strategii wykrywania włamań

Serwery przypisaliśmy ich usługom, bastiony są maksymalnie wzmocnione, a reguły zapór sieciowych są zaaplikowane. Nadszedł czas na skonfigurowanie systemu NIDS. Przypomnijmy sobie twierdzenia z podrozdziału „Określenie zakresu odpowiedzialności".

Zadajmy sobie ponownie to pytanie. Za co jestem odpowiedzialny? Czy jest to podsieć, konkretny serwer czy tylko określona usługa na konkretnym porcie w wybranym serwerze — należy skupić się dokładnie na tym, co mamy chronić.

System NIDS monitorujący serwer może być zainstalowany na tym serwerze lub na innej stacji pod warunkiem, że cały ruch kierowany do serwera jest widziany z tej stacji. Umieszczenie systemu NIDS na tym samym komputerze, który ma być monitorowany, jest najprostszym podejściem i zapewnia dostęp do odpowiedniego ruchu sieciowego.

Umieszczenie systemu NIDS na innym serwerze daje jednak następujące korzyści.

- Procesy związane z działaniem systemu NIDS nie będą miały wpływu na działanie monitorowanego serwera. W przeciwnym razie zalew dziwnych pakietów spowoduje, że system NIDS będzie z przejęciem generował statystyki, rysował wykresy i dostarczał raporty, a dla właściwej funkcji serwera zabraknie czasu. Jest to zagadnienie, które należy wziąć pod rozwagę, planując wykorzystanie zaawansowanej konfiguracji systemu NIDS.
- Gdy monitorowany system przejmie włamywacz, system NIDS działający na nim może zostać wyłączony lub, co gorsza, również zmodyfikowany przez włamywacza w taki sposób, że będzie generować raporty wskazujące na prawidłową pracę.

W każdym razie wydaje się, że najlepszym miejscem umieszczenia systemu NIDS jest ta sama podsieć w stosunku do routera i zapory sieciowej, w której znajduje się monitorowany system. Gdy wykorzystany jest tylko jeden NIDS, powinien on monitorować cały ruch przechodzący przez najbliższy router. Gdy wykorzystane są dwa systemy NIDS, lub jeden z dwoma czujnikami, jeden z nich powinien monitorować ruch za routerem i zaporą, a drugi przed nimi. W ten sposób będziemy widzieć, co się dzieje za zaporą i czy zapora i router nadal wykonują swoją pracę prawidłowo.

Oczywiście należy monitorować ruch do każdego z chronionych systemów i z niego, co sprowadza nas ponownie do zakresu odpowiedzialności. Z drugiej strony, monitoring powinien obejmować jak największy możliwy zakres zagadnień. Pakiet SYN na port `21/tcp` może być pomyłką. Pakiety SYN na port 21 wszystkich maszyn w sieci (a nawet na nieużywanych adresach) z całą pewnością są skanowaniem sieci. Z nieżyczliwymi intencjami.

Zatem nawet gdy administrator konfigurujący system NIDS nie jest odpowiedzialny za wszystkie systemy w sieci, z całą pewnością powinien wiedzieć, kiedy jeden z nich wygląda na przejęty przez napastnika. Ten system może w takim przypadku zacząć monitorować ruch w sieci na rzecz napastnika. Po określeniu miejsca umieszczenia systemu NIDS należy ustalić zakres zagadnień, które będzie kontrolować.

Ustalanie reguł

W tym podrozdziale zajmiemy się następującymi tematami.

- Detektory włamań nie wykrywają włamań.
- Pozytywne specyfikacje są złe.
- Negatywne specyfikacje nie są lepsze.
- Heurystyczne wykrywanie anomalii — Święty Graal.

Detektory włamań nie wykrywają włamań

Już o tym wspominaliśmy. Więc powtórzymy, bo to ważne. „Wykrywanie włamań" jest prawdopodobnie najlepszym określeniem, które zostało wymyślone przez dostawców tego typu rozwiązań po prostu po to, by klienci zrozumieli, o co chodzi. W najlepszym przypadku systemy IDS wykrywają ataki, a nie udane włamania. W najgorszym — systemy IDS wykrywają podejrzany ruch, który przypomina atak. Większość systemów IDS zaledwie sprawdza dane i nagłówki pakietów w poszukiwaniu wzorców znanych ataków i raportuje takie dopasowania. Należy pamiętać o tym, czytając wyniki pracy systemu IDS.

Pozytywne specyfikacje są złe

Wiemy, że niektóre pakiety oznaczają kłopoty. Każdy pakiet z jednocześnie ustawionymi znacznikami SYN i FIN oznacza złe wieści. Jeśli pakiet SYN zawiera dane, coś się popsuło w sieci. Pakiety SSH z doklejonym na końcu tekstem */bin/sh* lub znanymi sekwencjami kodów uruchamiających powłokę (*shellcode*) oznaczają próbę wywołania przepełnienia bufora (z drugiej strony istnieje pewne drobne prawdopodobieństwo, że zakodowane dane będą miały postać przypominającą *shellcode*).

Pozytywne specyfikacje działają tak. Gdy bierzemy udział w okazaniu i przyglądamy się szeregowi osób w celu wybrania osoby podejrzanej, przypominamy sobie plakaty osób ściganych. Ktoś, kto jest w całym kraju ścigany za przestępstwa, jest podejrzany z definicji.

Główną wadą pozytywnych specyfikacji jest to, że najpierw muszą być utworzone. A więc jeśli napastnik użyje nieznanego wcześniej ataku lub nawet wystarczająco zmodyfikuje typowy atak, nie zostanie wykryty przez IDS.

Negatywne specyfikacje nie są lepsze

Najpierw zajęliśmy się łatwą stroną — określeniem rzeczy złych. Teraz nadszedł czas na odrobinę kreatywnego myślenia. Określmy rzeczy, które ogólnie mogą być w porządku, lecz w danej sytuacji nie powinny występować.

- **Czy którykolwiek z serwerów w sieci udostępnia usługę telnet?** Jeśli nie, połączenia na port `23/tcp` nie powinny mieć miejsca.
- **Czy zapora sieciowa blokuje port 25/tcp na wszystkich komputerach oprócz serwera poczty?** Jeśli tak, to ruch na tym porcie na innych maszynach oznacza kłopoty.
- **Czy zachęcasz użytkowników do wymiany pirackiego oprogramowania (tzw. warez) i obrotu plikami MP3 na serwerze NFS?** Nie? Skąd zatem wziął się ruch między tym serwerem i portem `6667` na *irc.copyright.violation.net*?
- **Czy w organizacji wykorzystuje się wyłącznie Linuksa?** Skąd zatem tyle ruchu protokołu NetBIOS (usługi sieciowe MS Windows) w sieci?

Są to specyfikacje negatywne. Aby je znaleźć, należy najpierw nakreślić dokładny szkic oczekiwanego ruchu w sieci, a następnie zacząć poszukiwać w niej czegokolwiek, co nie pasuje do tego obrazka.

Heurystyczne wykrywanie anomalii — Święty Graal

Jak wspomnieliśmy, należy ciągle monitorować sieć w poszukiwaniu rzeczy, które nie powinny się w niej pojawiać. Niektóre fakty jednak stają się oczywiste po dłuższym czasie. Jak na przykład Krzyś z marketingu, który wylogowuje się z systemu codziennie o 16:30, bo śpieszy na Happy Hours do swojej ulubionej restauracji. Nie ma możliwości poznania tego wzorca z góry, chyba że ktoś monitoruje ruch w sieci przez długi okres czasu.

Ktoś powie: „Nic wielkiego, napiszę system NIDS, który będzie zapamiętywał wzór zdarzeń, a następnie raportował wszelkie odstępstwa od niego. Będę sławny!". Pomysł nie jest zły, lecz do tej pory znany jest jeden system realizujący tego typu funkcje w sposób skuteczny prawie w każdym środowisku. Ten system każdy z nas nosi pomiędzy uszami. Używaj tego systemu! Oglądaj od czasu do czasu ruch w sieci. Porównuj go do wpisów w dziennikach systemowych. Przyzwyczaj się do wzorców i jeśli to możliwe wymyśl sposób konfiguracji systemu NIDS w taki sposób, aby reagował na odstępstwa od nich.

Jeśli jednak ktoś znajdzie sposób na automatyzację tego procesu, potrzebny mu będzie agent handlowy — w takim wypadku chętnie służymy.

W następnym podrozdziale przyjrzymy się przykładowej implementacji mechanizmu NIDS, czyli programowi Snort.

Przykładowy system NIDS — Snort

Snort jest tak prosty, elastyczny i skuteczny, jakim sami go uczynimy. Ma niewielkie wymagania dotyczące pamięci i obciążenia procesora. Ponadto jest darmowy. Trudno nie docenić takich cech.

Użytkownicy lubiący kod źródłowy zapraszamy na stronę *www.snort.org/dl*, gdzie znajdą najnowszą wersję.

Opis i historia

Snort pojawił się na scenie sieciowej podczas prelekcji na konferencji *LISA* (*Large Installation System Administration*) w listopadzie 1999 roku. Marty Roesch odpowiedział na zapotrzebowanie, tworząc system NIDS nieco bardziej funkcjonalny od programu *tcpdump*, lecz mniejszy i tańszy od wszystkich innych. Od tej pory Snort rozwija się systematycznie bez zwiększania komplikacji instalacji, konfiguracji i użytkowania.

Choć Snort posiada wiele opcji, można używać go w najprostszej konfiguracji, wcześniej przez kilka minut trzeba poczytać dołączony do niego plik README. Marty założył firmę o nazwie Sourcefire, która zajmuje się rozwojem i sprzedażą specjalizowanych dodatków do systemu Snort. Choć wielu z nas może wykorzystywać Snort w takiej postaci, w jakiej jest dostępny, to wielkie organizacje mają zupełnie inne potrzeby. Komercyjne firmy, jaką jest Sourcefire, są w stanie takie potrzeby zrealizować.

Przegląd funkcji systemu Snort (wybrane opcje wiersza poleceń)

Snort posiada sporo opcji wiersza poleceń. Przyjrzyjmy się najważniejszym z nich, które pozwolą skłonić ten system do pracy.

Snort może śledzić pakiety na interfejsie sieciowym lub odczytywać je ze zrzutu w pliku. Chcemy monitorować sieć. Opcja `-i interfejs` wskazuje systemowi Snort nazwę interfejsu sieciowego, na którym ma śledzić pakiety, najczęściej interfejsem tym będzie eth0. Jeśli nie jesteśmy pewni, pomocny może okazać się program *netstat -in*, który wypisze listę aktywnych interfejsów sieciowych.

Wynik działania programu Snort można przekierować do katalogu dziennika. Domyślnie Snort stara się uporządkować dane pakietów w strukturze katalogów we wskazanym katalogu dziennika. Każdy z podkatalogów w tym katalogu odpowiada adresowi IP i zawiera pliki odpowiadające portom źródłowym, docelowym i protokołom. Jeśli Snort nie jest uruchomiony na systemie-pułapce (tzw. *honeypot*), kiedy taki poziom szczegółowości jest pożądany, można wyłączyć ten mechanizm za pomocą opcji `-N`.

Snort może generować ostrzeżenia w oparciu o określone w konfiguracji reguły dopasowania — to jest główne działanie, którego oczekujemy od systemu NIDS. Domyślnie pliki lądują w pliku w katalogu dziennika, który w domyślnej konfiguracji znajduje się w */var/log/snort/*. Opcja `-l katalog` pozwala zmienić położenie katalogu dziennika. Opcja `-s` spowoduje zapisywanie ostrzeżeń w pliku `/var/log/secure`.

Reguły wykorzystywane przez Snort zapisywane są w plikach. W głównym pliku z regułami można wykorzystać instrukcję ładującą inny plik z regułami, dzięki czemu reguły można wygodnie organizować w osobne pliki. Opcja `-c plik_regul` wskazuje systemowi główny plik z regułami.

Zalecamy uruchamianie systemu Snort z wykorzystaniem katalogu zawierającego następujące podkatalogi:

- `./logs`: zawierający ostrzeżenia w pliku *alerts*,
- `./rules`: zawierający wszystkie reguły, w tym główny plik reguł,
- `Snort_Command`: plik zawierający pełny wiersz poleceń uruchamiający system Snort z odpowiednimi opcjami.

Przykładem może być następujące polecenie:

```
snort -i eth0 -N -l ./logs -c ./rules/rules_root
```

Polecenie to uruchamia system Snort w trybie nasłuchu na porcie eth0, wykorzystujący reguły z katalogu *./rules* i zapisujący ostrzeżenia w katalogu *./logs*.

Pełny wykaz opcji wiersza poleceń można uzyskać, wykorzystując opcję `-h`.

Reguły i więcej reguł

Dobry podręcznik reguł systemu Snort można znaleźć pod adresem *www.snort.org/docs/writing_rules*. Właściwy podręcznik nosi nazwę *Snort User's Manual*. Nie mamy zamiaru uzupełniać go w niniejszej książce, lecz przedstawimy kilka prostych reguł będących modyfikacjami przykładowych z dystrybucji systemu Snort. Oto pierwsza z nich:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN SYN FIN";flags:SF;)
```

Powyższa reguła generuje ostrzeżenie w wyniku napotkania pakietu SYNFIN. Zmienne `$EXTERNAL_NET` oraz `$HOME_NET` stanowią adresy podsieci w notacji CIDR zdefiniowane w pliku reguł. Drugie pole zawiera nazwę protokołu. Następne jest wyrażenie w postaci:

```
<adres źródłowy> <port źródłowy> -> <adres docelowy> <port docelowy>
```

Alternatywna forma dopasowuje pakiet przechodzący w dowolnym kierunku:

```
<adres> <port> <> <adres> <port>
```

Reszta reguły (w nawiasach) ma następujący format:

```
opcja:argument;[opcja:argument;]
```

Opcja `msg` ustawia treść komunikatu zapisywanego przy dopasowaniu reguły. Komunikat zostanie zapisany do dziennika wraz ze znacznikiem czasu (*time stamp*), adresem i portem źródłowym i docelowym oraz z innymi informacjami na temat pakietu.

Opcja `flags` definiuje znaczniki, które musi posiadać pakiet, aby wywołać ostrzeżenie. W tym przypadku dopasowany zostanie pakiet zawierający znaczniki `SYN` i `FIN`.

Zwróćmy uwagę na zmienne `$HOME_NET` oraz `$EXTERNAL_NET`. Snort, podobnie jak większość innych systemów NIDS, najlepiej działa, gdy jest dopasowany do środowiska. Ustawienie tych zmiennych w pliku konfiguracyjnym systemu Snort pozwoli wykrywać ataki na systemy w sieci lokalnej i uniknąć fałszywych alarmów.

Kolejny przykład:

```
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"SHELLCODE linux shellcode";
➥content:"|90 90 90 e8 c0 ff ff ff|/bin/sh";)
```

Opcja `content` w tej regule wyszukuje dane w pakiecie pasujące do ciągu znaków w argumencie. Liczby pomiędzy liniami pionowymi (`|`) określają ciąg liczb szesnastkowych, każda para cyfr określa jeden bajt. Fragment */bin/sh* stanowi już zwykły tekst. Przedstawiona reguła wyszukuje w danych znany *shellcode* używany do uzyskania uprawnień konta *root* w systemie. Ta reguła dopasuje każdy pakiet IP (zarówno protokołu TCP, jak i UDP).

Następnie możemy wypróbować regułę wykorzystującą konkretny adres w sieci. Załóżmy, że serwer FTP działa w naszej sieci tylko pod adresem 10.20.400.15. Każdy ruch FTP na inny adres jest uważany za podejrzany, więc dodajemy następującą regułę:

```
alert tcp any any -> !10.20.400.15 21 (msg:"Podejrzany pakiet FTP ";)
```

W tym przykładzie przemyciliśmy ważną możliwość. Znak wykrzyknika (!) przed adresem oznacza „każdy adres inny od 10.20.400.15". Znak wykrzyknika jest bardzo użyteczny podczas definiowania specyfikacji negatywnych, które omówiliśmy wcześniej. Gdyby dana maszyna obsługiwała wyłącznie serwer FTP, można by zastosować następującą regułę:

```
alert tcp any any -> 10.20.400.15 !21 (msg:"Potencjalny skan serwera FTP";)
```

W tym momencie doceniłeś już zalety prostoty. Serwer dedykowany umożliwia zastosowanie znacznie prostszych reguł wykrywania ataków!

Wtyczki: przetwarzanie wstępne i końcowe (portscan)

Snort został zaprojektowany tak, aby łatwo było dodawać do niego nową funkcjonalność za pomocą wtyczek (*plugins*) przetwarzających dane przed obróbką ich przez Snort lub po ich wykorzystaniu. Wtyczki definiuje się w plikach reguł i, jak ma to miejsce w przypadku większości zagadnień związanych z systemem Snort, składnia wykorzystania wtyczek opisana jest w pliku *snort.conf* dostarczanym w dystrybucji systemu Snort.

Zalecamy stosowanie szczególnie wtyczki *portscan*. Pozwala na prostą i szybką identyfikację skanowania portów przez odnajdywanie charakterystycznych pakietów skierowanych jednocześnie na wiele portów (można zdefiniować liczbę pakietów i okres czasu), a następnie zapisuje wyniki w określonym pliku. Plik wynikowy działania tej wtyczki warto zapisać w katalogu *./logs* lub utworzyć specjalny katalog *./scans* służący w tym celu.

Podsumowanie

W niniejszym rozdziale omówiliśmy systemy NIDS jako integralną część strategii obronnej w sieci. Przed opisaniem zagadnień systemów NIDS rozpatrzyliśmy użyteczne strategie sieciowe, takie jak wykorzystanie dedykowanych serwerów, konfigurację zapór sieciowych i użycie dedykowanych hostów bastionowych w podsieci DMZ. Następnie omówiliśmy systemy NIDS w świetle projektu sieci i pożądanej lokalizacji tego systemu. Zarówno zakres działania, jak i lokalizacja systemu NIDS ma wpływ na reguły monitorowania ruchu sieciowego zdefiniowane w tym systemie. Nie uda się wykorzystać wszystkich zalet stosowania systemu NIDS, zapory sieciowej i innego oprogramowania wspomagającego bezpieczeństwo, jeśli będziemy je traktować jak „magiczne pudełka" czarodziejskim sposobem rozwiązujące wszystkie problemy. Najlepsze wyniki działania mechanizmów obronnych uzyskamy wtedy, gdy każdy z jego elementów będzie uzupełniać i wspomagać działanie drugiego, a nad wszystkim będzie panował administrator.

Następnie opisaliśmy zagadnienia projektowania strategii bezpieczeństwa uwzględniającej wykorzystanie systemu NIDS. Częścią planu było wprowadzenie do systemu Snort i pokazanie zagrożeń w sieci, które czyhają na nasze systemy każdego dnia. Innym celem było przedstawienie strategii bezpieczeństwa jako współpracujących elementów obejmujących kontrolę dostępu do sieci, dedykowane serwery i system NIDS. Naszym celem było również zaciekawienie czytelnika, ponieważ jesteśmy pewni, że w tej chwili konfigurujesz swój system NIDS, zamiast czytać te słowa. Owocnej parcy ze Snortem!